НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.415.2

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Метод динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувачів»

Виконав: студент ІІ курсу, групи КП-71мн
Коломієць Іван Валерійович _____
(підпис)

Керівник: доцент кафедри ПЗКС, к.т.н.,
Олещенко Любов Михайлівна _____
(підпис)

Рецензент: доцент кафедри автоматики та управління
в технічних системах ФІОТ, к.т.н., доцент
Полторак В.П. _____
(підпис)

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., Онай М. В. _____
(підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2017 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Коломійцю Івану Валерійовичу

1. Тема дисертації «Метод динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувачів», науковий керівник дисертації Олещенко Любов Михайлівна, к.т.н., доцент кафедри ПЗКС, затверджені наказом по університету від «8» квітня 2019 р. №1075-С

2. Термін подання студентом дисертації «17» травня 2019 р.

3. Об'єкт дослідження: процес програмної організації пошуку інформації в мережі Інтернет.

4. Предмет дослідження: методи інформаційного пошуку на основі адаптації процесу пошуку до індивідуальних особливостей користувачів інформаційно-пошукових систем.

5. Перелік завдань, які потрібно розробити:
- провести аналіз існуючих методів пошуку інформації сучасних інформаційно-пошукових систем;
- дослідити існуючі автоматизовані системи пошуку інформації;
- розробити та дослідити модифікований метод на адаптації процесу пошуку до індивідуальних особливостей користувачів;
- обґрунтувати вибір критеріїв оптимізації модифікованого методу;
- розробити автоматизовану систему проектування пошукового сервісу;
- проаналізувати ефективність модифікованого методу в порівнянні з сучасними на основі визначених критеріїв оптимізації.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
- теоретичні аспекти побудови та оптимізації модифікованого методу динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувачів;
- схема роботи модифікованого методу динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувачів;

- діаграми з результатами щодо обґрунтування критеріїв ефективності методу;
- схема алгоритму функцій ідентифікації контенту;
- особливості архітектури програмного комплексу;
- часові характеристики роботи автоматизованої системи підтримки прийняття рішень на основі модифікованого методу.

7. Орієнтовний перелік публікацій:
   - Стаття “The Method of Dynamic Design of a Search Engine Based on Automated Analysis of User Requests”
   - Тези доповіді “Метод динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувача”

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС		

9. Дата видачі завдання «11» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	16.12.2017	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	17.03.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	14.05.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення; підготовка матеріалів доповіді на конференції ПМК-2018	16.10.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	21.12.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	22.02.2019	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу	11.04.2019	
8.	Оформлення текстової і графічної частини магістерської дисертації	08.05.2019	

Студент І.В. Коломієць

Науковий керівник дисертації Л.М. Олещенко

РЕФЕРАТ

Актуальність теми. Сьогодні спостерігається експонентне зростання кількості джерел інформації у світі, що зумовлено збільшенням чисельності її споживачів, обсягу створеної і доступної інформації. Це викликає все більші складнощі в ефективному пошуку інформації, що є наслідком, з одного боку, особливостей людино-машинної взаємодії, а з іншого – семантичної неоднорідності джерел інформації. Розв’язання цієї проблеми полягає в індивідуалізації засобів інформаційного пошуку, тобто в адаптації процесу пошуку до індивідуальних особливостей користувачів, що дозволить швидко знаходити релевантну інформацію з мінімальними зусиллями користувачів. З погляду мінімізації часу і вартості пошуку найбільш перспективним є використання моделей користувачів для розширення запиту під час проведення адаптивного інформаційного пошуку, що дозволяє суттєво скоротити час інтерактивної взаємодії і витрат користувачів, оскільки уточнення запиту виконується на стороні клієнтської частини.

Об’єктом дослідження є процес програмної організації пошуку інформації в мережі Інтернет.

Предметом дослідження є методи інформаційного пошуку на основі адаптації процесу пошуку до індивідуальних особливостей користувачів інформаційно-пошукових систем.

Мета роботи: покращити релевантність інформації в пошуковій видачі та надати пошуковий сервіс на основі адаптації процесу пошуку інформації до індивідуальних особливостей користувача.

Методи дослідження. Для розроблення методу динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувачів використано метод PRF з показником релевантності, метод праймінгу, методи математичної статистики та методи об'єктно-орієнтованого програмування.

Наукова новизна полягає у розробленні методу, який, на відміну від існуючих методів пошуку інформації в мережі Інтернет, дозволяє збільшити швидкість виконання ранжирування пошукової видачі на 10.2%. Запропоновано новий підхід для пошуку інформації, який дозволяє зменшити витрати дискового простору пошукової системи на 17.61% за рахунок відмови від індексування веб-сторінок, яке використовується в сучасних пошукових системах.

Практична цінність отриманих у роботі результатів полягає в тому, що запропонований метод динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувачів дозволив збільшити швидкість видачі релевантної інформації, завдяки чому можливо зменшити затримку в роботі пошукової системи.

Здійснено програмну реалізацію запропонованого методу, що може бути використана для пошуку релевантної інформації та налаштування роботи з цією інформацією.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (м. Київ), опублікована стаття «The Method of Dynamic Design of a Search Engine Based on Automated Analysis of User Requests» в міжнародному фаховому виданні "Electronics and Control Systems" (2018 р.).

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів, висновків, списку літератури та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень.

У першому розділі розглянуто загальні підходи до вирішення задач пошуку інформації, була виконана оцінка методів існуючих сучасних інформаційно-пошукових систем.

У другому розділі запропоновано математичний апарат для розроблення нового методу інформаційного пошуку в мережі Інтернет; описано основну ідею методу.

У третьому розділі запропоновано засоби реалізації удосконаленого методу інформаційного пошуку; описано загальний алгоритм методу.

У четвертому розділі наведено оцінку ефективності методу динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувачів; оцінку якості ранжування пошукової видачі інформації; наведено порівняння отриманих результатів з результатами аналогів; виконано оцінку швидкодії методу; швидкодія методу порівняна з швидкодією інших розглянутих у роботі методів пошуку інформації.

У висновках наведено отримані результати роботи.

У додатках наведено фрагменти програмної реалізації запропонованого методу та копії графічних матеріалів.

Робота виконана на 85 аркушах, містить 2 додатки та посилання на список використаних літературних джерел з 52 найменувань. У роботі наведено 20 рисунків та 5 таблиць.

Ключові слова: інформаційно-пошукова система, пошуковий сервіс, користувач, пошуковий запит, релевантність інформації, ранжування, вартість пошуку, індексування веб-сторінок, метод PRF, метод праймінгу.

ABSTRACT

Theme urgency. Nowadays there is an exponential growth in the number of information sources in the world, due to the increase in the number of its customers, the amount of generated and accessible information. This is of increasing difficulties within the efficient search for information, which is a consequence, on the one hand, of the features of human-machine interaction, and on the other hand - the semantic heterogeneity of the sources of information. The solution to this problem is to individualize the means of information retrieval, that is, in the adaptation of the search process to the individual user peculiarities, which will allow to find qiuckly relevant information with minimal effort of users. In terms of minimizing time and cost of search, the most promising is the use of user patterns to expand the query while conducting an adaptive information search, which significantly reduces the time of interactive collaboration and user spending, since the query refinement is performed on the client side.

The object of research is the process of software organization of information search on the Internet.

The subject of the research is information search methods based on the adaptation of the search process to the individual characteristics of users of information retrieval systems.

The purpose of the research: to improve the relevance of information in the search results and provide a search service based on the adaptation of the information retrieval process to individual user characteristics.

Research methods. To develop the method of dynamic design of the search service on the basis of the adaptation of the search process to the individual characteristics of users, the PRF method with the indicator of relevancy, the method of priming, methods of mathematical statistics and methods of object-oriented programming are used.

The scientific novelty of the work is the method has developed that, unlike the existing methods of search of information on the Internet, allows you to increase the speed of execution of ranking search results by 10.2%.

A new approach to finding information that reduces the cost of disk space of a search engine by 17.61% has suggested on the basis of the rejection of indexing web pages, which is used in modern search engines.

Practical value of the results obtained in the work is that, the proposed method of dynamically designing a search service on the basis of the adaptation of the search process to the individual user features has allowed to increase the speed of the issuance of relevant information, which can reduce the delay in the search engine.

The software realization of the proposed method is implemented, which can be used to find relevant information and how to work with this information.

Approbation. The main provisions and results of the work were presented and discussed at the scientific conference of masters and postgraduates "Applied Mathematics and Computer", PMK-2018 (Kyiv), , the article " The Method of Dynamic Design of a Search Engine Based on Automated Analysis of User Requests " was published in the international professional edition " Electronics and Control Systems" (2018).

Structure and content of the thesis. The master's dissertation consists of an introduction, four sections, conclusions, list of references and appendices.

The introduction provides a general description of the work, an assessment of the current state of the problem is carried out, the relevance of the direction is substantiated researches.

In the first section general approaches to solving information search problems using modern information retrieval systems are considered; an evaluation of the methods of existing search engines was performed.

In the second section a mathematical device was proposed for the development of a new method of information retrieval in the Internet; the main idea of the method was described.

In the third section the means of realization of the advanced method of information search have offered; the general algorithm has described and the program realization of this method has given.

In the fourth section the estimation of efficiency of the received method of dynamic designing of the search service on the basis of adaptation of the search process to the individual peculiarities of users and quality assessment of the ranking of search information has given; the comparison of the results with the results of analogues has given; evaluation of the performance of the method is performed; the performance of the method is comparable to the speed of other information search methods considered in the work.

The conclusions are given the results of work.

The appendixes contain fragments of software implementation of the proposed method and copies of graphic materials.

The work is performed on 85 sheets, contains 2 attachments and a link to the list of used literary sources of 52 titles. The paper presents 20 figures and 5 tables.

Key words: information retrieval system, search service, user, search query, relevance of information, ranking, search cost, indexing of web pages, PRF method, priming method.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ІПС – інформаційно-пошукова система.

PRF – Pseudo relevance feedback, це метод обробки пошукового запиту інформаційно-пошукових систем.

Показник релевантності – це міра відповідності результатів пошуку завданню, поставленому в пошуковому запиті.

API – Application Programming Interface, це опис способів, набір класів, процедур, функцій, структур або констант, за допомогою яких одна комп'ютерна програма може взаємодіяти з іншими програмами.

Кешування – швидкісна пам'ять або частина оперативної пам'яті, де зберігаються копії часто використовуваних даних. Забезпечує швидкий доступ до них. Кеш-пам'ять зберігає вміст і адреси даних, до яких часто звертається процесор.

ПМ – пошукова машина, це програмно-апаратний комплекс, сервер, API, на якому виконується пошуковий рушій.

SEO – Search Engine Optimization, виконує внутрішню і зовнішню оптимізацію сайту з метою підвищення позиції сайту в списку сторінок, знайдених пошуковими системами за конкретними запитами.

Метадані – це дані, що характеризують або пояснюють інші дані.

SERP – Search Engines Result Page, це сторінка пошукового сайту, яка відкривається у відповідь на певний пошуковий запит, тобто результат роботи пошукової системи.

Вартість пошуку – це термін, що означає ефективність пошуку, складається із швидкості надання інформації, релевантності та повноти цієї інформації.

Релевантність – це ступінь відповідності знайденого документа або набору документів інформаційним потребам користувача.

IP-адреса – це унікальний числовий номер мережевого рівня, який використовується для адресації комп'ютерів чи пристроїв у мережах.

Ранжування – це сортування сайтів в пошуковій видачі.

DSMS – Data Stream Management System, являє собою програмну систему для управління безперервними потоками даних.

SLA – Service Level Agreement, це угода між постачальником послуг і користувачем, яка містить кількісні та якісні характеристики наданих послуг, такі як їх доступність, підтримка користувачів, час виправлення та інше

CSRF – Cross Site Request Forgery це тип веб-атаки, що призводить до виконання певних дій від імені користувача на веб-сторінці.

Праймінг – це явище роботи пам'яті, яке представляє собою зміну швидкості або точності рішення задачі, що спостерігається після пред'явлення інформації.

Polyfill – бібліотека, яка реалізує підтримку стандартів, де замовчуванням підтримка цих стандартів частково або повністю відсутня.

DOM – Document Object Model це специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами.

REST – Representational State Transfer, це підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

ВСТУП

Обсяг інформації у світі зростає з експонентною швидкістю, це зумовлено збільшенням обсягу доступної інформації, створенням нової інформації та зростанням кількості користувачів у мережі Інтернет.

Це робить все більші складнішим пошук релевантної інформації, що є наслідком, з одного боку, неоднорідності джерел інформації, а з іншого – особливості людино-машинної взаємодії. Для розв’язання даної проблеми потрібно в алгоритмі інформаційного пошуку зосередити увагу на індивідуалізації пошукового запиту до особливостей користувача, тобто зводити процес пошуку до індивідуальних потреб користувачів, що дає можливість швидко знаходити потрібну інформацію за короткий час із мінімальними зусиллями.

У роботі аналізуються алгоритми побудови сучасних інформаційно-пошукових систем та пропонується удосконалений метод динамічного пошуку з урахуванням індивідуальних особливостей користувачів, який дозволяє збільшити вартість пошуку.

Створено сервіс для розроблення власної пошукової системи, у якому алгоритм ранжування замінено на алгоритм сортування пошукових систем за категоріями методу PRF. Інформація заповнюється системою послідовних об'єктів, яка представляє собою список. Об'єкти містять ключову інформацію, за допомогою цих ключів можна налаштувати роботу з інформацією і підвищити актуальність пошуку.

Сервіс містить графічні інструменти для оновлення, зберігання і видалення інформації у пошуковій системі. Аналіз конфігурації служби пошуку виконується для збільшення вартості пошуку для кінцевого користувача. Наведено результати дослідження, де зазначаються недоліки пошукової системи розробника цієї системи і особливості генерування документації з її використання на основі налаштувань для кінцевого користувача.

Система сортування аналізує запит кінцевого користувача і надає знайдену інформацію, відсортовану за релевантністю. Розроблений метод забезпечує прискорення роботи алгоритму пошуку інформації інформаційно-пошукової системи та забезпечення якості інформації, знайденої для індивідуальних потреб користувачів.

У роботі використовуються елементи теорії графів та алгоритмів, теорії надійності, метод апроксимації для оцінки показників релевантності інформації. Наведені результати експериментів, порівняння характеристик інформаційно-пошукових систем.

1. АНАЛІЗ МЕТОДІВ ОРГАНІЗАЦІЇ ПОШУКУ ІНФОРМАЦІЇ В СУЧАСНИХ ІНФОРМАЦІЙНО-ПОШУКОВИХ СИСТЕМАХ

1.1. Загальні алгоритми пошуку інформації в мережі Інтернет

На даний час відомі такі пошукові системи, як Google, Yandex, Baidu і багато інших. Алгоритми пошукових систем унікальні для кожної пошукової системи і так само важливі, як і ключові слова.

Алгоритм пошукової системи – це набір правил або унікальна формула, яку пошукова система використовує для визначення значимості документів, і кожна пошукова система має свій власний набір правил [1]. Ці правила визначають, чи є документ релевантним або корисним для користувача, чи містить він важливу інформацію, яка буде корисна для користувача, і безліч інших функцій для ранжування і виведення списку результатів для кожного розпочатого пошукового запиту інформаційного пошуку. Алгоритми пошуку є різними для кожної пошукової системи і ретельно засекречуються, але є певні критерії, спільні для усіх алгоритмів пошукових систем.

Першим з них є релевантність. Релевантність перевіряє алгоритм пошукової системи на корисну інформацію або документ. Це може бути сканування ключових слів або аналіз використання цих ключових слів. Алгоритм визначає, чи має цей документ корисну інформацію для конкретного ключового слова у запиті. Розташування ключових слів також є важливим фактором для забезпечення релевантності пошуку. Документи з ключовими словами в заголовку, або перших кількох рядках тексту будуть краще ранжуватися за цим ключовим словом, ніж документи, які не мають цих функцій. Частота ключових слів також важлива для релевантності. Якщо ключові слова з'являються часто, але не є результатом заповнення ключових слів, документ буде мати більш високий рейтинг [2].

Наступним критерієм пошуку є індивідуальні фактори. Друга частина критеріїв роботи алгоритмів пошукової системи – це фактори, які відрізняють цю конкретну пошукову систему від будь-якої іншої

пошукової системи. У кожної пошукової системи є свої унікальні алгоритми, і окремі фактори цих алгоритмів пояснюють, чому пошуковий запит в Google показує різні результати, відмінні від Yandex або Baidu. Одним з найбільш поширених індивідуальних факторів є кількість сторінок, які індексує пошукова система. Вони можуть індексувати більше сторінок або індексувати їх частіше, але це може дати різні результати для кожної пошукової системи. Деякі пошукові системи також карають за спам, в той час як інші ні [3].

Ще є декілька факторів, які є складовою алгоритмів, як і раніше, вони є індивідуальні для кожної пошукової системи, це фактори положення сторінки. Факторами положення сторінки є вимір кліків і зв'язувань. Частота кліків і посилань може бути показником того, наскільки релевантний документ для реальних користувачів і відвідувачів, і це може привести до того, що алгоритм підвищить рейтинг документу.

PageRank (PR) – це алгоритм, який використовується Google для ранжування веб-сторінок в результатах пошуку. PageRank був названий на честь Ларрі Пейджа, одного із засновників Google. PageRank – це алгоритм вимірювання важливості сторінок сайту. За даними Google, PageRank працює шляхом підрахунку кількості та якості посилань на сторінку, щоб визначити приблизну оцінку важливості веб-сайту. Основне припущення полягає в тому, що більш важливі веб-сайти можуть отримувати більше посилань з інших веб-сайтів [4].

На даний час PageRank – не єдиний алгоритм, який використовується Google для упорядкування результатів пошуку, це перший алгоритм, який був використаний компанією Google, і він є найвідомішим.

PageRank – це алгоритм аналізу посилань, який присвоює числову вагу кожному елементу набору документів з гіперпосиланнями, наприклад, World Wide Web, з метою «вимірювання» його відносної важливості у наборі [5] (рис. 1.1).

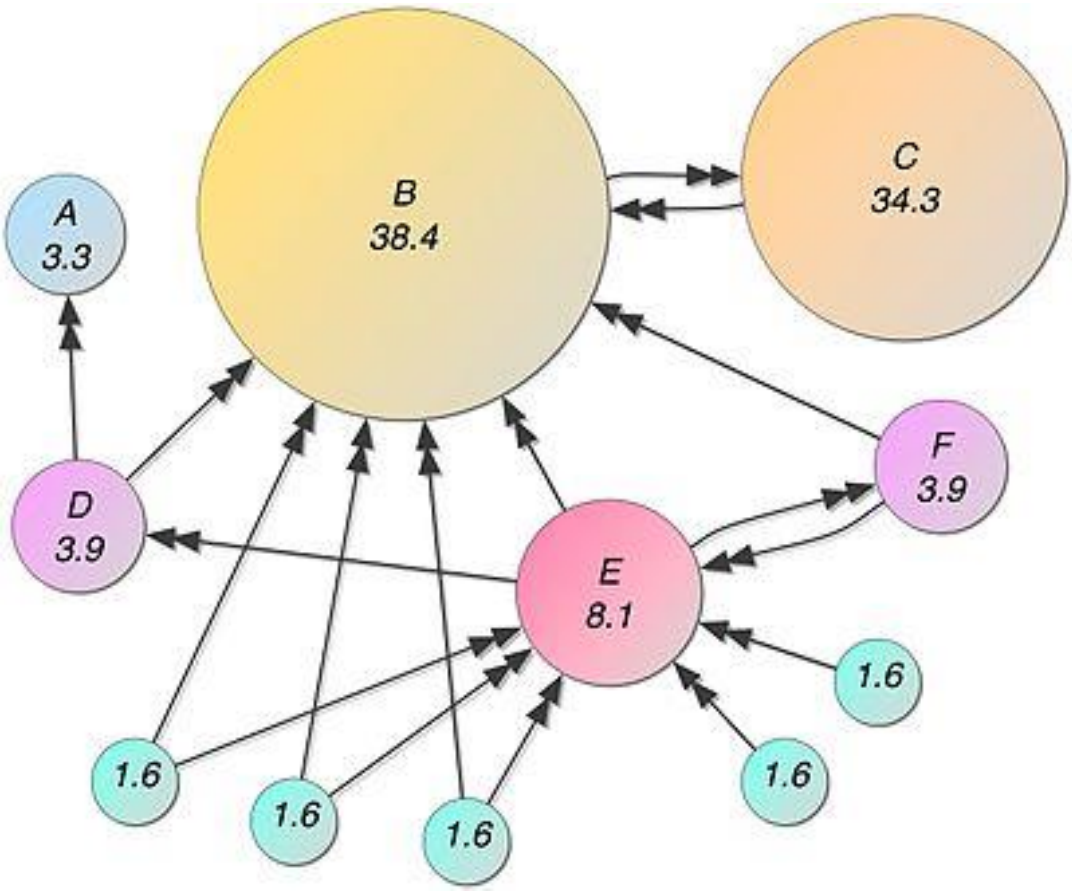


Рис. 1.1. Графічне зображення роботи алгоритму PageRank [6]

Алгоритм пошуку може бути застосований до будь-якої колекції об'єктів з взаємними цитатами і посиланнями. Числова вага, яку він призначає будь-якому елементу E, називається PageRank of E і позначається як PR (E). Інші чинники, такі як оцінка, можуть сприяти важливості об'єкта [7].

Утворюється PageRank з математичного алгоритму, заснованому на webgraph, створений на всіх World Wide Web сторінках, як вузли та гіперпосилання, беручи до уваги концентраторів влади, таких як cnn.com або USA.gov. Значення рейтингу вказує на важливість конкретної сторінки. Гіперпосилання на сторінку вважається голосуванням підтримки [8].

PageRank – сторінка визначається рекурсивно і залежить від кількості і метрики PageRank усіх сторінок, які посилаються на неї («вхідні посилання»). Сторінка, на яку посилаються багато сторінок з високим PageRank, сама отримує високий рейтинг.

Численні наукові статті, що стосуються PageRank, були опубліковані з моменту публікації Пейджа і Бріна [9]. На практиці концепція PageRank може бути вразливою для маніпуляцій.

Були проведені дослідження, згідно з якими виявлення помилкового запиту значно впливає на рейтинг PageRank [10]. Мета полягає в тому, щоб знайти ефективні способи ігнорування посилань з документів з помилковим впливом.

PageRank веб-сторінки залежить від наступних факторів [11].

1. Частота і місце розташування ключових слів на веб-сторінці. Якщо ключове слово з'являється тільки один раз у тілі сторінки, воно отримає низьку оцінку для цього ключового слова.
2. Як довго існує веб-сторінка: люди створюють нові веб-сторінки кожен день, і не всі вони довго залишаються на місці. Надається більше значення сторінкам з історією.
3. Кількість інших веб-сторінок, які посилаються на цю сторінку: перевіряється, скільки веб-сторінок посилаються на певний сайт, щоб визначити його релевантність.

Посилювальні алгоритми – це алгоритми, які використовуються із основним алгоритмом пошуку, їх реалізація може бути різною, але усі вони вирішують певні проблеми в пошуку [12].

Розглянемо декілька прикладів цих алгоритмів [13].

1. Відповіді з графу знань. У 2012 році компанія Google зробила запуск графіку знань, який налічує понад один мільярд реальних людей, місць і речей з більш ніж 50 мільярдами фактів і зв'язків між ними. Світ складається з реальних речей, а не просто текстових рядків. Тому потрібно будувати графік знань, щоб показати, як все пов'язано. За допомогою даного алгоритму можна отримати швидкі відповіді.

2. Напрямки за допомогою трафіка. Завжди було очевидно, що коли люди шукали в посиланні адреси, вони не хотіли посилатися на сайти, де згадуються вулиці. Швидше за все, вони хотіли знати шлях, як туди дістатися. Тому був створений алгоритм напрямку на посилання за допомогою трафіку.
3. Прямі відповіді. Іноді потрібні прямі відповіді на певні запити, тому потрібно об'єднуватися з компаніями, які можуть надавати інформацію та послуги, ліцензувати їх вміст, щоб надавати корисні відповіді безпосередньо на сторінці результатів пошуку. Наприклад, для пошуку розкладу показів фільмів у місцевому кінотеатрі даний алгоритм дозволяє співпрацювати з постачальниками даних, які мають у своєму розпорядженні актуальну і достовірну інформацію про те, коли фільми показуються у заданому районі та з постачальниками послуг з продажу квитків.
4. Вибрані фрагменти. При заданому запиті мета даного алгоритму – допомогти користувачу швидко і легко знайти відповідь. Вибрані фрагменти допомагають швидко отримати відповіді на питання, привертаючи увагу до програмно генерованих фрагментів з веб-сайтів, які алгоритми вважають доречними до конкретного заданого питання. Усі рекомендовані фрагменти містять фрагмент інформації, цитованої зі стороннього веб-сайту, а також посилання на сторінку, заголовок сторінки і URL-адресу.
5. Багаті списки. Найкращою відповіддю на питання є не завжди один об'єкт, а список або група пов'язаних людей, місць або речей. Тому, коли користувач буде шукати «каліфорнійські маяки» або «знаменитих жінок-астрономів», даний алгоритм покаже йому список цих речей у верхній частині сторінки.

6. Відповіді, перш ніж запитати. Користувачі очікують, що інформація у них під рукою. Ось чому спочатку потрібно показувати мінімальний набір інформації, яку користувач переглядає кожен день. Наприклад, теперішній час, погоду у місті, курс валют тощо. Це дозволяє отримати інформацію без посилання запиту, що дуже зручно [14].

1.2. Аналіз сучасних інформаційно-пошукових систем

Пошукова система Google є потужним інструментом, без якого було б практично неможливо знайти потрібну інформацію при перегляді веб-сторінок. Google використовує спеціальний алгоритм для генерації результатів пошуку. Google ділиться лише загальними фактами про свій алгоритм. Це допомагає Google залишатися конкурентоспроможною пошуковою системою [15].

Google використовує автоматизовані програми, які називаються Павуками або сканерами. Також як і інші пошукові системи, Google має великий індекс ключових слів. Що відрізняє Google, так це те, як він ранжує результати пошуку, що, в свою чергу, визначає порядок, в якому Google відображає результати на своїй сторінці пошукової системи Search Engine Results Page. Google використовує алгоритм PageRank, який присвоює кожній веб-сторінці оцінку релевантності [16].

Дуже важливим фактором роботи PageRank в алгоритмі пошуку Google є кількість інших веб-сторінок, які посилаються на цю сторінку: перевіряється, скільки веб-сторінок посилаються на певний сайт, щоб визначити його релевантність.

Це простіше зрозуміти на прикладі. Розглянемо пошук за терміном «планета Земля». У міру того, як все більше веб-сторінок посилаються на сторінку Планети Земля в Discovery, рейтинг сторінки Discovery збільшується. Коли сторінка Discovery займає більш високе місце, ніж інші сторінки, вона відображається у верхній частині сторінки результатів

пошуку Google. Оскільки Google розглядає посилання на веб-сторінку як голосування, обманути систему нелегко, але можливо [17].

Компанія Google розпочала експеримент зі своєю пошуковою системою у 2008 році. Вперше Google дозволяє групі бета-тестерів змінювати порядок ранжування результатів пошуку. В цьому експерименті бета-тестери можуть рекламувати або знижувати результати пошуку і адаптувати свій досвід пошуку так, щоб він був більш актуальним [18].

Розглянемо, як Google визначає, що потрібно показувати в результатах пошуку. Алгоритм починає свою роботу ще до того, як користувач набере свій запит у поле вводу (рис. 1.2).

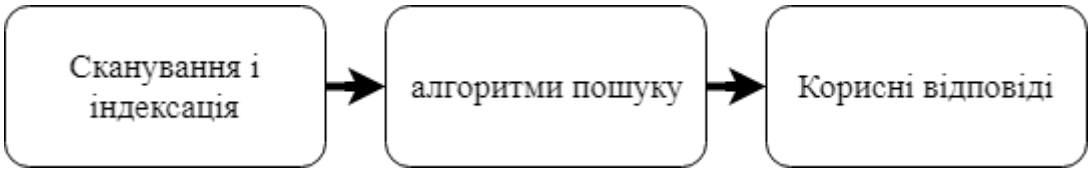


Рис. 1.2. Схема роботи пошукового сервісу

Спочатку відбувається сканування та індексація. Google використовує веб-сканерів та Павуків для організації інформації з веб-сторінок та іншого загальнодоступного контенту в пошуковому індексі. Потім працюють алгоритми пошуку. Системи рейтингу Google сортують сотні мільярдів веб-сторінок у пошуковому індексі, щоб за лічені секунди отримати корисні і релевантні результати. Наступнім етапом є алгоритми пошуку.

Системи ранжування складаються з ряду алгоритмів, які аналізують те, що шукає користувач, і яку інформацію потрібно повернути.

Схема процесу ранжування пошуку інформації, якими користується Google для надання корисної інформації з Інтернету зображена на рис. 1.3.

Аналіз запиту, розуміння значення пошуку має вирішальне значення для отримання точних результатів пошуку. Тому, щоб знайти сторінки з відповідною інформацією, потрібно спочатку проаналізувати, що означають слова у пошуковому запиті. Будуються мовні моделі, щоб спробувати розшифрувати, які рядки слів потрібно шукати в індексі. Це

включає в себе такі кроки, як інтерпретація орфографічних помилок, ідентифікація типу запиту, застосовуючи деякі з останніх досліджень в області розуміння природної мови.

Рис. 1.3. Схема процесу пошуку та отримання інформації в мережі Інтернет

Наприклад, система синонімів допомагає пошуку зрозуміти запит, навіть якщо слово має кілька значень.

Пошук веб-сторінок, забезпечує пошук веб-сторінки з інформацією, що відповідає запиту. Коли виконується пошук на базовому рівні, алгоритм шукає пошукові терміни в індексі, щоб знайти відповідні сторінки. Алгоритм аналізує, як часто і де ці ключові слова з'являлися на сторінці, в заголовках або в тексті. Крім зіставлення ключових слів, алгоритм шукає докази, щоб виміряти, наскільки добрі потенційні результати пошуку дають користувачам те, що вони шукають. Коли робиться пошук, наприклад, виразу «оперативна пам’ять», користувач, ймовірно, не хоче, щоб сторінка зі словом «пам’ять» з’явилась у нього сотні разів. Алгоритм намагається з'ясувати, чи містить сторінка відповідь на запит, а не просто повторити цей запит. Таким чином, алгоритм пошуку аналізує, чи містять сторінки релевантний контент, наприклад, зображення літаків, відео або список архітекторів. Нарешті, алгоритм перевіряє, чи написана сторінка на тій же мові, що і запитання, щоб визначити пріоритетність сторінок на вибраній мові.

Ранжування корисних сторінок. Для типового запиту існують тисячі, навіть мільйони веб-сторінок з потенційно актуальною інформацією. Щоб у першу чергу оцінити кращі сторінки, використовується алгоритм для оцінки корисності цих веб-сторінок. Цей алгоритм аналізує сотні різних факторів, щоб спробувати розкрити найкращу інформацію, яку може запропонувати пошук, від свіжості контенту до частоти появи пошукових термінів, а також від того, наскільки зручна сторінка. Щоб оцінити достовірність і авторитет у своїй тематиці, алгоритм шукає сайти, оцінює дані користувачів із схожими запитами. Якщо інші сайти за темою посилаються на сторінку, це ознака того, що інформація високої якості. В Інтернеті є багато сайтів зі спамом, які намагаються пробитися до вершини результатів пошуку за допомогою таких методів, як повторення ключових слів знову і знову або через посилання, що проходять через PageRank. Ці сайти надають дуже поганий користувацький досвід і можуть навіть зашкодити або ввести в оману користувачів [19].

Огляд контексту, такої інформації, як місце розташування користувача, історія пошуку в минулому і налаштування пошуку, допомагають алгоритму адаптувати результати так, щоб інформація була найбільш корисно і актуально для користувача в даний момент. Алгоритм використовує дані про країну і місце розташування користувача для доставки контенту відповідно його регіону. Наприклад, якщо користувач перебуває в Чикаго і шукає «матч», швидше за все, він отримає результати з американського футболу і «матчів Чикаго». Якщо шукати «матч» в Лондоні, алгоритм оцінить результати про футбол. У деяких випадках алгоритм також може персоналізувати результати, використовуючи інформацію про недавній пошуковий запит користувача. Наприклад, якщо користувач шукає «Барселона» і недавно шукав «Барселона проти Арсеналу», це може бути важливою підказкою, що йому потрібна інформація про футбольний клуб, а не про місто.

Отримання кращих результатів пошуку, перш ніж алгоритм представить результати. Алгоритм оцінює, як вся відповідна інформація поєднується: чи є одна тема серед результатів пошуку чи їх багато, а також занадто багато сторінок, присвячених одній вузькій інтерпретації. Алгоритм прагне надати різноманітний набір інформації в форматах, які найбільш корисні для конкретного типу пошуку.

Завдяки більшій кількості контенту і більшого розмаїття в Інтернеті, ніж будь-коли раніше, Google має можливість пропонувати результати пошуку в широкому діапазоні форматів, щоб допомогти користувачу швидко знайти потрібну інформацію, але нерідко це важко зробити через дуже великий діапазон та час на пошук релевантної інформації [20].

Мережа Інтернет постійно розвивається, кожну секунду публікуються сотні нових веб-сторінок. Це відображено в результатах пошуку, Google постійно аналізує інформацію у мережі для індексації нового контенту.

Останнім кроком алгоритму є корисні відповіді. Залежно від запиту користувача деякі сторінки результатів швидко змінюються, а інші стають більш стабільними. Наприклад, коли користувач шукає останній рахунок спортивної гри, потрібно виконувати найсвіжіші оновлення, в той час як результати про історичну тематику можуть залишатися незмінними роками.

Сьогодні Google обробляє трильйони пошукових запитів щороку. Кожен день 15% запитів, які обробляються, це ті, які ми ніколи раніше не бачили. Побудова алгоритмів пошуку, які можуть надати найбільш корисні результати для всіх цих запитів, є складним завданням, яке вимагає постійного тестування якості та інвестицій.

Тисячі інженерів і учених працюють над вдосконаленням алгоритмів і пошуком нових корисних способів пошуку інформації [21]. Тільки у 2016 році було покращено близько 1600 рішень у пошуку Google, який використовує наступні алгоритми пошуку інформації.

1. Відповіді з графа знань.
2. Напрямки за допомогою трафіку.
3. Прямі відповіді.
4. Вибрані фрагменти.
5. Багаті списки.
6. Відповіді, перш ніж запитати [22].

Пошукова система Yandex відповідає на запити користувачів відповідними веб-документами, знайденими в Інтернеті. Проте розмір Інтернету в даний час розраховується в екзабайтах – квінтильйонах або мільярдах мільярдів байтів інформації. Зрозуміло, що пошукова система не переглядає цю величезну кількість даних кожен раз, коли він відповідає на новий пошуковий запит [23].

Для пошуку система Yandex використовує пошуковий індекс, який представляє собою базу даних усіх слів і їх розташування, відомих пошуку. Розташування слова – це поєднання його положення на веб-

сторінці і адреси веб-сторінки в Інтернеті. Пошуковий індекс схожий на глосарій або телефонний довідник. На відміну від глосарію, який містить тільки обрані терміни, пошуковий індекс реєструє кожне слово, яке коли-небудь зустрічалось у пошуку. На відміну від телефонної книги, у якій перераховані імена і адреси, пошуковий індекс має більше однієї зареєстрованої адреси для кожного слова.

Пошукова система в мережі Інтернеті працює у два етапи. По-перше, вона сканує мережу, зберігаючи свою копію на своїх серверах. По-друге, вона відповідає на пошуковий запит користувача, отримуючи відповідь від своїх серверів [24].

Перш ніж пошукова система зможе почати пошук, вона повинна підготувати інформацію, яку знаходить в Інтернеті, для пошуку. Цей процес називається індексацією. Спеціальна комп'ютерна система – веб-сканер регулярно переглядає Інтернет, завантажує нові веб-сторінки і обробляє їх. Це створює свого роду «точну копію» Інтернету, яка зберігається на серверах пошукової системи і оновлюється після кожного сканування [25].

У Yandex є два сканери – один з них, основний сканер, індексує всі веб-сторінки, які він зустрічає, а інший, відомий як Orange, виконує експрес-індексацію, щоб гарантувати, що найактуальніші документи, в тому числі ті, які з'явилися в Інтернеті, або навіть секунди до сканування, доступні в індексі пошукової системи. Обидва сканери мають списки очікування веб-сторінок, які необхідно проіндексувати. У списки постійно додаються нові посилання, які сканери знаходять на відвідуваних ними сторінках. Нові посилання також можуть з'явитися в списках очікування після того, як власники сайтів додадуть свої сторінки в індекс за допомогою сервісу системи. Адміністратори веб-сайту також можуть надати додаткову інформацію, наприклад, як часто оновлюється їх веб-сайт тощо [26].

Перед початком процесу сканування спеціальна програма – планувальник або алгоритм «Павук-мандрівник» створює розклад, відповідно до якого відвідуються веб-сторінки. Планування грунтується на ряді чинників, необхідних для пошуку інформації, таких як популярність посилань або частота оновлення сторінки. Після складання розкладу, інший компонент пошукової системи – Павук вступає у дію (рис. 1.4). Павук регулярно відвідує сторінки за розкладом. Якщо веб-сайт доступний для павука і функціонує, програма завантажує сторінки веб-сайту відповідно до графіку. Павук аналізує код і мову завантаженого документа і потім відправляє цю інформацію на сервери для зберігання [27].

На сервері зберігання інша програма очищує документ від непотрібної інформації розмітки, залишаючи тільки текст. Потім вона витягує інформацію про місцезнаходження кожного слова і додає всі слова в цьому документі в покажчик. Вихідний документ також зберігається на сервері до наступного сканування. Це дозволяє алгоритму пропонувати своїм користувачам можливість переглядати документи, навіть якщо сайт тимчасово недоступний. Якщо сайт закривається або документ видаляється або оновлюється, алгоритм видаляє його зі своїх серверів або замінює його більш новою версією.

Індекс пошуку разом з копіями усіх проіндексованих документів, включаючи їх тип, код і мову, утворює базу даних пошуку (рис. 1.5). Щоб йти в ногу з постійно мінливою природою інтернет-контенту і бути впевненим, що пошукова система може знайти останню і найактуальнішу інформацію у відповідь на призначений для користувача пошуковий запит, пошукова база даних повинна регулярно оновлюватися. Перш ніж пошукова система зможе знайти і повернути результати кінцевим користувачам, кожне нове оновлення бази даних спочатку відправляється на сервери «основного пошуку». Базові пошукові сервери містять тільки основну частину бази даних пошуку без спаму, дзеркальних сайтів або

інших, які не відносяться до суті документів. Це частина пошукової бази даних, яка безпосередньо відповідає на запити користувачів [28].

Оновлення бази даних пошуку відправляються з серверів зберігання на сервери основного пошуку у вигляді «пакетів» один раз в декілька днів. Це дуже ресурсномісткий процес. Щоб знизити навантаження на сервери, дані передаються вночі – коли пошуковий трафік знаходиться на найнижчому рівні. Нові частини бази даних порівнюються з використанням ряду параметрів з останньою версією, доступною під час попереднього сканування, щоб переконатися, що оновлення не псує якість результатів пошуку. Після успішної перевірки якості стара версія замінюється останнім оновленням [29].

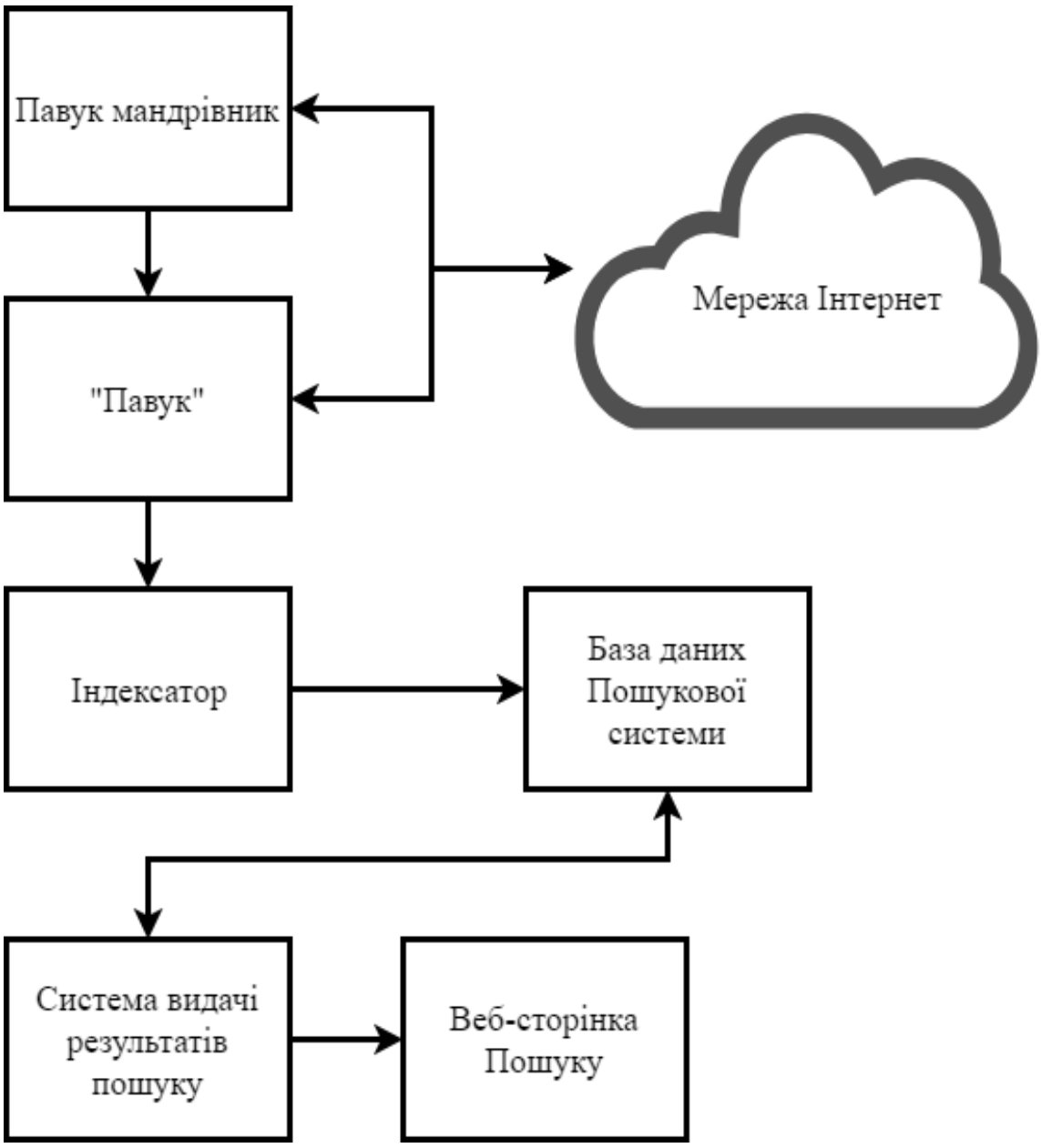


Рис. 1.4. Схема роботи алгоритму «Павук»

Сканер Orange призначений для пошуку інформації в реальному часі. І його планувальник, і Павук налаштовані на пошук останніх документів і вибірку з величезної кількості сторінок, які можуть представляти інтерес для користувача. Ці документи обробляються миттєво і відправляються на основні пошукові сервери. Оскільки кількість цих документів відносно невелика, оновлення може відбуватися в режимі реального часу навіть протягом дня без ризику перевантаження серверів [30].

Пошукова система працює у два етапи. На першому етапі відбувається сканування мережі, індексування сторінок, підготовка їх до

пошуку. Другий етап – пошук відповіді на конкретний користувальницький запит в раніше створеній пошуковій базі даних [31].

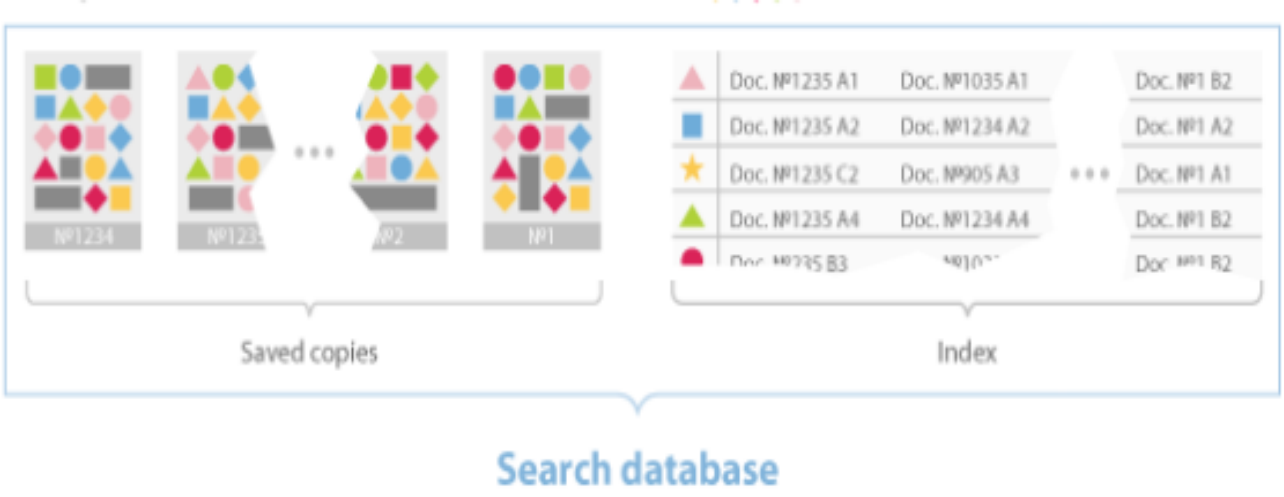


Рис. 1.5. Фрагмент бази даних індексів з копіями усіх проіндексованих документів

Зростання інтернет-інформації приводить до того, що важко отримати корисну інформацію. Як ефективно отримати доступ до цієї інформації і використовувати її, є основним питанням у роботі пошукової системи. Будучи джерелом інформації для всієї пошукової системи, система збору даних в основному відповідає за збір, зберігання і оновлення інформації в Інтернеті. Вона рухається по мережі, як Павук, тому її називають «Павуком». Наприклад, кілька поширених пошукових Павуків називаються Baiduspdier, Googlebot, Sogou Web Spider [32].

Система сканування «Павук» є важливою гарантією для джерела даних пошукової системи: якщо мережа розуміється як орієнтований граф, робочий процес Павука можна розглядати як обхід орієнтованого графа.

Починаючи з деяких важливих вихідних посилань, користувач може постійно відкривати нові джерела посилань і сканувати гіперпосилання в документі, щоб захопити якомога більше цінних документів. Для великої системи, наприклад, як Baidu, кожен раз, коли існує ймовірність зміни, видалення або появи нового посилання на документ, сторінка, яку система

вже просканувала, оновлюється, підтримуючи бібліотеку документів та посилань та є дуже складною задачею.

Алгоритм пошуку в Baidu працює за наступними принципами. Величезна кількість інтернет-ресурсів вимагає, щоб система обходу використовувала швидкість пропускання якомога ефективніше і захоплювала якомога більше цінних ресурсів при обмежених ресурсах сервісу та швидкості пропускання. Це створює ще одну проблему, яка викликана пропускною спроможністю для документу. Якщо ступінь занадто велика, це безпосередньо вплине на нормальну поведінку доступу користувача. Отже, в процесі сканування необхідний певний контроль для досягнення мети, щоб не тільки вплинути на призначений для користувача доступ до документу, але і для захопити якомога більше цінних ресурсів.

Використовуваний код повернення – кілька кодів повернення, підтримуваних Baidu [33].

1. Найбільш поширений код 404 означає, що документ не знайдено. Якщо термін дії веб-сторінки закінчився, вона зазвичай видаляється в бібліотеці. У короткостроковій перспективі, якщо алгоритм знову знайде дане посилання, документ не буде сканувати ся.
2. Код повернення 503 означає, що служба недоступна і вважає, що документ тимчасово недоступний, зазвичай веб-сайт тимчасово закритий, а пропускна здатність обмежена. Алгоритм не буде видаляти це посилання безпосередньо і буде відвідувати його кілька разів за короткий час. Якщо документ був відновлений, він знову буде нормально сканувати ся, якщо він продовжить повертати 503, то це посилання буде видалено у базі.
3. Код повернення 403 означає заборону і вважає, що сторінка в даний час заблокована. Якщо це нове посилання, алгоритм не буде переходити за посиланням деякий час і відвідає його

кілька разів за короткий термін, якщо він вже включений, алгоритм не буде видаляти його і до нього буде звертатися кілька разів.

4. Представник 301 коду означає, перехід назавжди на інше посилання. Алгоритм вважає сторінки перенаправлені на нове посилання.

Наступним етапом алгоритму є визначення кількох перенаправлень посилань. Деякі веб-сторінки в Інтернеті мають статус перенаправлення посилання з різних причин. Щоб правильно обробити цю частину ресурсів, алгоритм повинен ідентифікувати і оцінити перенаправлення посилань і запобігти шахрайству.

Алгоритм Baidu бере участь в скануванні мережевого протоколу. Пошукова система Baidu має складну стратегію сканування. Насправді, між пошуковими системами і постачальниками ресурсів існують взаємозалежні відносини. Пошукова система потребує налаштування для надання ресурсів для неї, інакше пошукова система не зможе задовольнити вимоги користувача до пошуку [34].

Налаштування системи повинні забезпечувати просування свого контенту через пошукові системи для залучення більшої аудиторії. Система сканування безпосередньо зачіпає інтереси провайдерів інтернет-ресурсів.

Для того, щоб дозволити пошуковій системі і налаштуванням створити безпрограшну ситуацію, обидві сторони повинні дотримуватися певних специфікацій в процесі сканування, щоб полегшити обробку даних і стикування між двома сторонами. Норми, які дотримуються в цьому процесі, є співвідносні з мережевими протоколами.

Згідно статистичних даних, Google обробляє, у середньому, понад 40 тисяч пошукових запитів за секунду, очевидно, що величезна кількість людей використовує Google у своєму повсякденному житті. Але це не єдиний великий гравець у цій сфері. У багатьох країнах Східної Європи та

України компанія Google зіштовхується з жорсткою конкуренцією, такі пошукові сервіси, як Yandex і Baidu, є привабливою альтернативою для користувачів Інтернету. Незважаючи на те, що Yandex не дуже добре відомий за межами Росії, є четвертим найбільшим постачальником послуг у всьому світі і контролює понад 61% пошуків у російськомовних країнах, таких як Казахстан та Білорусь. Великою перевагою Yandex над Google є її глибоке розуміння російської мови та граматики, яка надає російськомовним користувачам значно більш точний пошук рідною мовою, ніж може запропонувати Google. Служба також надає свої власні послуги для конкурентів Google, таких як карти, переклади та зберігання у хмарі, а підтримка російської мови Search Engine Optimization значно перевищує підтримку американського гіганта.

Маючи вражаючий спектр різноманітних послуг, включаючи Wiki, систему обміну файлами і навіть власну платформу соціальних медіа, Baidu стає все більш популярним серед китайських користувачів мережі. В даний час вона контролює понад 80% ринку пошуку Китаю і має понад 70 мільйонів активних користувачів на мобільних пристроях щодня.

Природно, Baidu має набагато більше розуміння китайських ієрогліфів і граматики, ніж Google, і, таким чином, оптимізує китайський контент більш ефективно, ніж Google. У зв'язку з різними правилами Інтернету в різних країнах, при оптимізації вмісту для Baidu користувачам рекомендується використовувати експертів для локалізації вмісту [35].

Хоча Google як і раніше вважається чемпіоном, коли мова йде про пошукові системи, варто зазначити, що існують сотні мільйонів пошуків, які проходять через інші платформи, такі як згадані вище, і вони не повинні бути знехтувані, коли мова йде про створення глобального пошукової системи в Інтернеті. Розуміння мови та культури має безперечну перевагу для успіху пошукових систем у всьому світі.

1.3. Аналіз проблем існуючих методів пошуку інформації

Пошукові системи виконують дві основні функції: сканування і створення індексу, надання користувачам пошуку ранжируваного списку сайтів, які вони визначили найбільш актуальними. Щоб знайти релевантну інформацію, пошукова система повинна сканувати свій індекс веб-сторінок на предмет змісту, пов'язаного з пошуком користувача. Пошукова система створює цей індекс, використовуючи програму «веб-сканер». Проблема полягає в тому, що пошуковому алгоритмі в системі потрібно постійно оновлювати інформацію в мережі Інтернет, але оскільки сам алгоритм не знає, що інформація оновилась на веб-сторінці, йому потрібно постійно робити сканування веб-сторінок.

Веб-сторінок в мережі Інтернет є дуже велика кількість, і у зв’язку із обмеженнями своєї обчислювальної здатності, пошукова система не може оновляти всі сторінки та підтримувати їх в актуальному стані. Цей алгоритм переглядає мережу Інтернет і зберігає інформацію про сторінки, які відвідує веб-сканер. Кожен раз, коли веб-сканер відвідує веб-сторінку, він робить її копію і додає свою адресу в індекс. Після цього веб-сканер переходить на всі існуючі адреси на сторінці, повторюючи процес копіювання та індексації. Потім він знову переходить на посилання в індексованих сторінках. Це є проблемою масштабування системи, сам алгоритм дуже багато витрачає ресурсів обчислювальної системи. Він продовжує робити це, накопичуючи величезний індекс з множини веб-сторінок.

Деякі веб-сайти не дозволяють сканерам відвідувати їх, це також є великою проблемою. Ці сторінки будуть виключені з індексу разом зі сторінками, на які ніхто не посилається. Інформація, яку збирає веб-сканер, потім використовується пошуковими системами. Це стає індексом пошукової системи. Кожна веб-сторінка, рекомендована пошуковою системою, переглядається сканером.

Пошукові системи сортують результати, щоб показати ті, які вони вважають найбільш корисними.

PageRank є найвідомішим алгоритмом, який використовується для покращення результатів веб-пошуку. Простіше кажучи, PageRank – це конкурс популярності веб-сторінок. Чим більше посилань вказують на веб-сторінку, тим більше корисною вона буде. Це означає, що вона буде знаходитися вище в результатах. Тут з'являються такі проблеми, що шанси знайти хоч щось корисне прирівнюються до нуля, або час є дуже великим, тому що пошукова система видає найпопулярніші відповіді, які, в свою чергу, не є актуальними.

Сторінки на першій сторінці результатів – ті, які PageRank вважає найкращими. Пошукові системи також звертають увагу на безліч інших «сигналів» при розробці замовлення, щоб показати користувачу результати. Наприклад, як часто сторінка оновлюється і надходить з надійного домену. Є багато пошукових систем на вибір. Різні пошукові системи використовують різні алгоритми. Це означає, що деякі сайти будуть давати свої результати в іншому порядку, або вони можуть навіть показати абсолютно різні результати [36].

Різноманітність алгоритмів пошуку приводить до того, що не можливо правильно налаштувати свою веб-сторінку так, щоб вона стала популярною в декількох пошукових сервісів, ця проблема полягає в тому що, пошукова система не здатна зрозуміти що потрібно користувачам. Щоб відповісти на ці питання, спочатку необхідно вивчити типи даних, які збираються і зберігаються в зв'язку з пошуками. Сервери пошукових систем в Інтернеті, такі як сервери кожної веб-служби, автоматично і систематично реєструють кожну подію, кожен запит сторінки. Журнал пошуку містить такі дані, як IP-адресу пристрою користувача, тип та мову використовуваного браузера, дату і час запиту, ідентифікатор файлу, встановленого у браузері користувача, і сам пошуковий запит.

IP-адреса (Інтернет-протокол) – це номер, який однозначно ідентифікує пристрій (комп'ютер, мобільний телефон тощо), що бере участь у комп'ютерній мережі та використовує Інтернет-протокол для

зв'язку. IP-адреса пристрою, підключеного до Інтернету, призначається постачальником послуг Інтернету (постійно) (статична IP-адреса) або періодично (динамічна IP-адреса).

Знаючи IP-адреси, постачальники пошукових систем можуть визначити інтернет-провайдера і приблизне географічне розташування використовуваного пристрою, але не можуть ідентифікувати людину, яка проводить пошук. Цю позицію можна порівняти зі становищем людини, яка знає домашню адресу іншої людини (місто, вулиця, номер будинку), але за відсутності місцевої карти ми не можемо знайти місце, де фактично знаходиться його будинок. Ця карта є тільки у інтернет-провайдера користувача, і провайдер може визначити, знаючи IP-адресу, дату і час запиту, інтернет-абонента, від якого надійшов запит. Але так само, як в одному будинку може бути більше однієї людини, до однієї IP-адреси може бути підключено більше одного комп'ютера і іншого пристрою, тобто більше одного користувача. Тому малоймовірно однозначно ідентифікувати користувача шляхом об'єднання даних, оброблених постачальником пошукової системи, і даних, оброблених інтернет-провайдером.

Через широке використання динамічних IP-адрес, їх можливості пов'язувати воєдино пошукові запити, запити веб-сторінок, що надходять від одних і тих же людей, також дуже обмежені. Оскільки постачальники пошукових систем, як і багато інших постачальників веб-послуг, потребують цієї можливості, наприклад, для збереження налаштувань і переваг, поліпшення їх послуг шляхом аналізу поведінки користувачів, вони також використовують файли ідентифікації.

Файл індетифікації – це невеликий файл даних, який встановлюється на комп'ютері користувача веб-сервером при першому відвідуванні користувачем веб-сторінки. При наступному відвідуванні сторінки в тому ж домені веб-браузер відправляє дані назад на сервер, визначаючи їх використання в якості повернення відвідувача. Без використання файлів

індетифікації кожен запит веб-сторінки буде незалежним від всіх інших, що призведе до відключення веб-служб, таких як веб-пошта та інтернет-магазини [37].

Термін дії деяких файлів індетифікації закінчується в кінці кожного сеансу, але постійні файли залишаються на жорсткому диску до закінчення терміну придатності. У багатьох випадках на веб-сторінці є частини, такі як рекламні оголошення, які належать іншому домену, і вони також можуть встановлювати так звані сторонні файли індетифікації на жорсткий диск комп'ютера користувача. Використовуючи їх, пошукові сервіси можуть відслідковувати рух користувачів з кількох веб-сайтів, які містять їх рекламу, що допомагає їм створювати профілі користувачів на основі звичок веб-серфінгу. Ці файли фактично ідентифікують веб-браузер користувача, а не самого користувача.

У поєднанні з IP-адресою ідентифікатор файлу, встановленого пошуковою системою, може пов'язувати різні пошукові запити з певним комп'ютером, але в деяких випадках неможливо визначити, хто стояв за ним, коли були зроблені запити. Пошукові системи зберігають також виконані пошукові запити разом з IP-адресою і ідентифікатором. У випадках, коли IP-адреса і дані дозволяють ідентифікувати користувача, ідентифікований користувач може бути пов'язаний з його пошуковими запитами, що може представляти реальну загрозу конфіденційності.

Пошукова система ідентифікації збирає, аналізує і зберігає дані для швидкого і точного пошуку інформації. Дизайн ідентифікаторів включає міждисциплінарні поняття з лінгвістики, та когнітивної психології, математики, інформатики для роботи із інформацією. Альтернативна назва процесу в контексті пошукових систем, призначених для пошуку веб-сторінок в Інтернеті, або веб-індексація [38].

Популярні пошукові двигуни фокусуються на повнотекстовій індексації онлайн-документів на оригінальній мові. Типи даних такі як медіа, відео, аудіо та графіка, також доступні для пошуку. Метадані

пошукової системи повторно використовують індекси інших сервісів або сторінок і не зберігають локальний індекс, тоді як ПМ на основі кешування постійно зберігають індекс разом із текстовою складовою сторінки. На відміну від повнотекстових індексів, сервіси з частковим текстом обмежують індексованих глибину, щоб зменшити розмір індексу. Більші служби зазвичай виконують індексацію в заздалегідь заданий інтервал часу через необхідний час витрат на обробку, тоді як агентські пошукові системи індексують в режимі реального часу.

Метою зберігання індексу є оптимізація швидкості і продуктивності при пошуку відповідних документів для пошукового запиту. Без індексу пошукова система буде сканувати кожен документ у базі, що потребує значного часу і обчислювальної потужності. Наприклад, хоча індекс в тисячу документів може бути запитаний протягом мілісекунди, послідовне сканування кожного слова в тисячі великих документів може зайняти кілька годин. Додаткове комп'ютерне сховище, необхідне для зберігання індексу, а також значне збільшення часу, необхідного для виконання оновлення, обмінюються на час, заощаджений під час пошуку інформації (рис. 1.6).

Ключовою функцією для пошукової системи є метод Фасета, оскільки він служить способом фільтрації даних за певним значенням атрибута. Хорошим прикладом цього аспекту є «бренди» на веб-сайті електронної комерції.

У Algolia кожен набір результатів для пошукового запиту містить найбільш поширені входження для кожного аспекту. Це означає, що якщо є багато значень, деякі з них будуть приховані [39].

Рішення цієї проблеми полягає у тому, щоб отримати більше значень та результатів, але цього може бути недостатньо, і користувачеві доведеться шукати значення вручну (рис. 1.7).

Індекс структури даних архітектури пошукових систем варіюються в залежності від способу виконання індексації та методів зберігання індексу, щоб відповідати наступним чинникам проектування.

Рішенням цієї проблеми була рекомендація розробникам пошукових систем створити новий індекс для кожного аспекту, який буде доступним для пошуку. Цей процес створював накладні витрати.

Рис. 1.6. Схема роботи пошукової системи на основі функцій ідентифікації контенту

При розробленні архітектури інформаційно-пошукової системи необхідно врахувати наступні фактори.

1. Фактори злиття – як дані потрапляють в індекс, або як слова або тематичні об'єкти додаються в індекс під час обходу текстового корпусу і чи можуть кілька індексаторів працювати асинхронно. Індексатор повинен спочатку перевірити, оновлює він старий контент або додає новий. Обхід зазвичай співвідноситься з політикою збору даних. Злиття індексу

пошукової системи за своєю суттю аналогічно команді злиття SQL і іншим алгоритмам злиття.

2. Методи зберігання – як зберігати дані індексу, тобто, чи повинна інформація бути стислою або відфільтрованою.
3. Розмір індексу – скільки пам'яті комп'ютера потрібно для підтримки індексу.
4. Швидкість пошуку – як швидко слово може бути знайдено в перевернутому індексі. Швидкість знаходження запису в структурі даних, у порівнянні з тим, як швидко вона може бути оновлена або видалена, є центральним завданням комп'ютерних наук.
5. Технічне обслуговування – як індекс підтримується з плином часу.
6. Відмовостійкість – наскільки важливо, щоб сервіс був надійним.

Проблеми пошуку включають в себе роботу з пошкодженням індексу, визначення того, чи можуть погані дані оброблятися ізольовано, роботу з несправним обладнанням, розбиття та схеми, такі як розбиття на основі хешу або складене розбиття, а також реплікація.

1. Суфікс дерево – структурований як дерево, підтримує лінійний пошук за часом. Побудований шляхом зберігання суфіксів слів. Дерево суфіксів є лінійним типом дерева. Спроби підтримують розширюване хешування, що важливо для індексації пошукової системи. Використовується для пошуку патернів у послідовності ДНК і кластеризації. Основним недоліком є те, що для зберігання слова в дереві може знадобитися простір, що перевищує простір, необхідний для зберігання слова. Альтернативним поданням є масив суфіксів, який вимагає менше віртуальної пам'яті і підтримує стиснення даних.

2. Інвертований індекс – зберігає список входжень кожного атомарного критерію пошуку, зазвичай у формі хеш-таблиці або дерева.
3. Індекс цитування – зберігає посилання або гіперпосилання між документами для підтримки аналізу цитування, предмет бібліографічних даних.
4. Ngram цитування – зберігає послідовності даних довжини для підтримки інших типів пошуку або аналізу тексту.
5. Документ-матриця – використовується в прихованому семантичному аналізі, зберігає входження слів в документах в двовимірній розрідженій матриці [40].

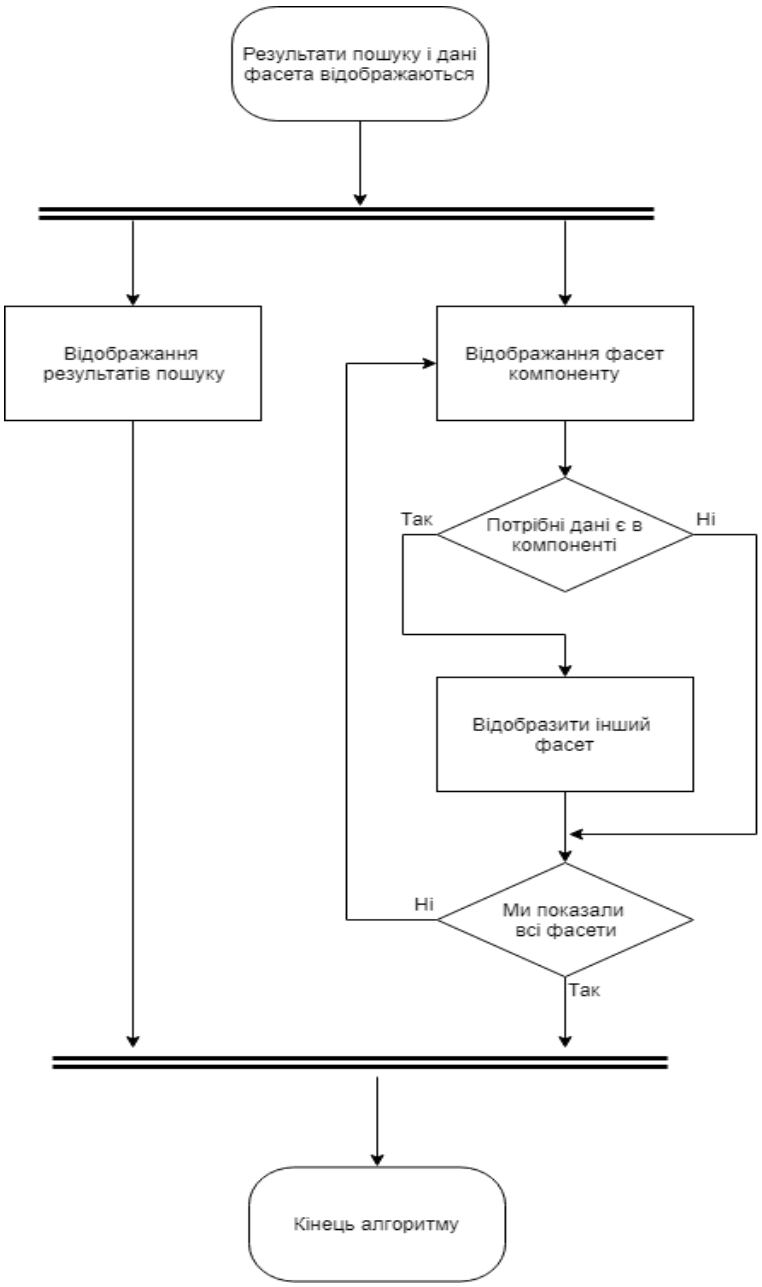


Рис. 1.7. Алгоритм роботи пошуку на основі метода Фасета

Індекс структури даних архітектури пошукових систем варіюється в залежності від способу виконання індексації та методів зберігання індексу, щоб відповідати критеріям проектування ІПС.

1.4. Висновки до розділу 1

У даному розділі розглянуто загальні методи пошуку інформації в сучасних інформаційно-пошукових системах, а саме PageRank, Павук, метод Фасета. Розглянуто проблеми існуючих методів пошуку інформації в мережі Інтернет.

Розглянуто підсилювальні алгоритми пошуку, які використовуються у будь-якій інформаційно-пошуковій системі, а саме відповіді з графу знань. Проаналізовано напрямки пошуку інформації за допомогою трафіка, прямий пошук та прямі відповіді, вибрані фрагменти, списки, відповіді перш ніж запитати.

Проблема сучасних пошукових методів полягає у тому, що вони витрачають велику кількість обчислювальних ресурсів на сканування веб-сторінок, а також оновлення цієї інформації. Іншою проблемою є значні витрати дискового простору, складність реалізації алгоритмів для сканування веб-сторінок та час їх виконання.

Підтримка алгоритмів у потрібному на теперішній час стані за стандартом WC3, робить даний спосіб роботи з інформацією незручним та невигідним в експлуатації. Для вирішення виділених проблем необхідно створити інформаційно-пошукову систему, яка б дозволила відмовитися від складних алгоритмів пошуку, а саме від Павука та індексатора, оскільки дані алгоритми потребують значного об’єму обчислювальних ресурсів. Оскільки з даною архітектурою пошуку на основі індексації веб-сторінок неможливо відмовитися від цих алгоритмів, потрібно сотворити власну архітектуру зберігання інформаційних даних в одній єдиній системі без використання посилань на веб-сторінки. Це забезпечить швидкість оновлення інформації, спростить складність алгоритмів пошуку та зберігання інформації, а також збільшить швидкість виконання пошукового запиту за допомогою зменшення витрат обчислювальних ресурсів системи на алгоритми пошуку та індексації веб-сторінок.

2. СТВОРЕННЯ УДОСКОНАЛЕНОГО МЕТОДУ ПОШУКУ ІНФОРМАЦІЇ В МЕРЕЖІ ІНТЕРНЕТ

2.1. Критерії для розроблення удосконаленого методу пошуку інформації в мережі Інтернет

Автоматична інтеграція даних за запитом з можливістю пошуку і запит інтегрованих даних може бути корисною у різних ситуаціях. Наприклад, це може бути короткострокова інтеграція даних з декількох алгоритмів. Також може використовуватися середньострокова інтеграція баз даних двох пошукових сервісів, де нова база даних, що містить всі дані, підлаштовується під першу, або довгострокова інтеграція персональних даних, наприклад, електронні магазини, які містять інформацію про товари, документи або медіа дані.

Різні контексти та проблеми релевантності інформації призводять до різних часів життя у просторі даних, тому було вирішено розбити їх на групи короткострокових, середньострокових та довгострокових баз даних (рис. 2.1).

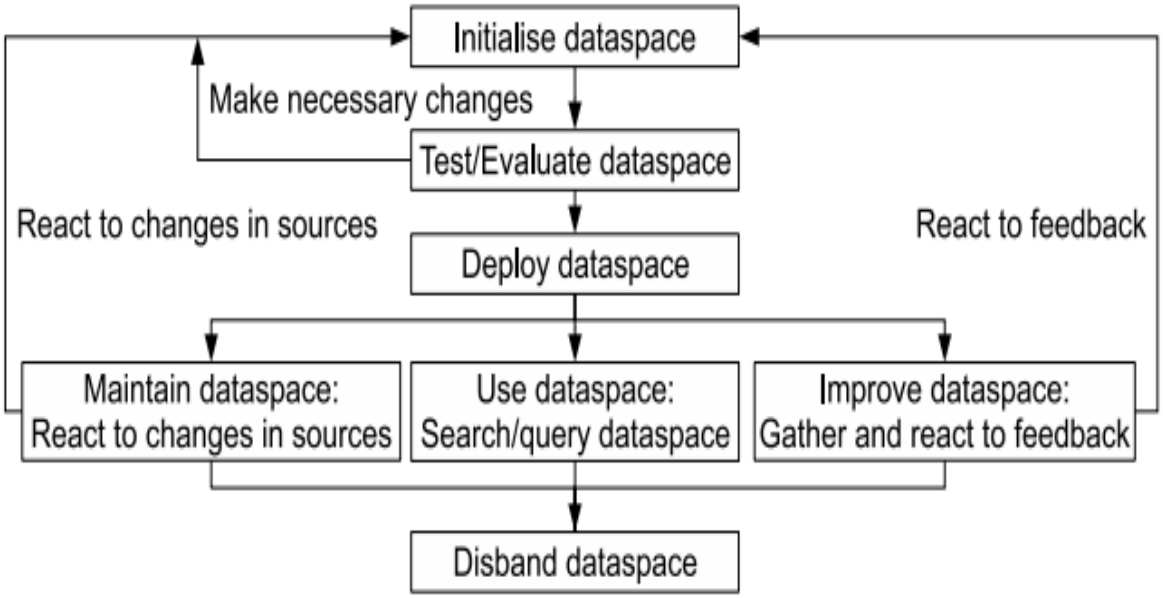


Рис. 2.1. Концептуальний життєвий цикл простору даних

Простору даних в різних контекстах може знадобитися тільки підмножина життєвого циклу. Розглянуті етапи перераховані в життєвому циклі з ініціалізацією, тестуванням, оцінкою фази розгортання, обслуговування, використання та поліпшення, що позначаються як init, test, depl, maint, use та impr відповідно.

Простір даних, як і будь-яке традиційне програмне забезпечення для інтеграції даних, ініціалізується. Ініціалізація може включати в себе ідентифікацію ресурсів даних, до яких здійснюється доступ та інтеграція цих ресурсів. За ініціалізацією може послідувати етап оцінки і тестування перед розгортанням. Етап розгортання може не потребувати багато часу, наприклад, в разі особистого розгортання шаблону простору даних, розташованого на одій обчислювальній системі, може включати надання доступу до простору даних для користувачів або переміщення інфраструктури простору даних на сервер.

Для ініціалізації DSMS переважно вимагає обмеженого зусилля, інтеграція може поліпшуватися з часом методом оплати в міру використання, поки він використовується для пошуку і запиту інтегрованих інформаційних даних. У постійно мінливих умовах, DSMS також повинна реагувати на зміни, наприклад, у змінах даних в базі данних, яка може вимагати підтримки для поступової інтеграції. Для прикладу на потік даних, які позначають переходи між фазами, а також ініціалізація, використання, обслуговування та етапи поліпшення (рис. 2.2).

Якість обслуговування відноситься до будь-якої технології, яка управляє трафіком даних, щоб зменшити втрати пакетів, затримки в мережі. Якість обслуговування контролює і управляє мережевими ресурсами, встановлюючи пріоритети для певних типів даних у мережі.

Корпоративні мережі повинні надавати передбачувані і вимірні послуги, такі як додатки, передача голосу, відео і дані, чутливі до затримки і проходять через мережу. Організації використовують якість

обслуговування для задоволення вимог трафіку програм, таких, як передача голосу і відео в реальному часі та для запобігання погіршення якості, викликаного втратою пакетів, затримкою і тремтінням.

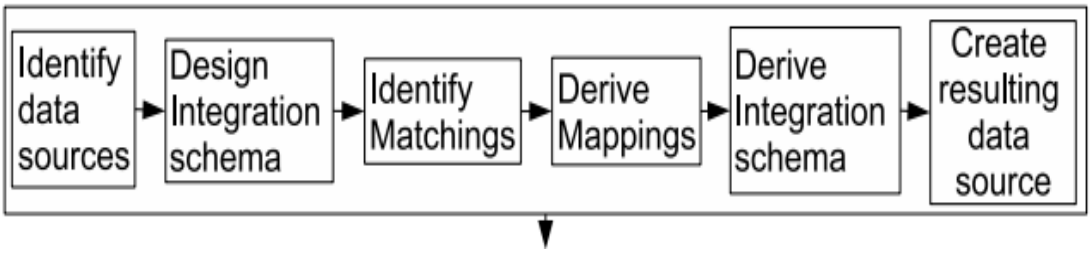


Рис. 2.2. Ініціалізація простору даних

Деякі атрибути якості обслуговування в табл. 2.1, з більш високими або меншими значеннями дають кращі результати окремо. Щоб відрегулювати процес за допомогою нормалізації, він ділиться на три типи: позитивні значення наприклад, репутація R (1), негативні значення, наприклад, час відгуку T і вартість виконання C (2), та процентні значення, наприклад, доступність A (3). Наступні рівняння використовуються для нормалізації позитивних, негативних і процентних значень відповідно:

$$QoS_R = \frac{x - q^{min}}{q^{max} - q^{min}}, \quad (1)$$

де QoS_R являє собою значення, отримане з порівняння репутації, x – значення якості даних, які необхідно порівнювати, q^{min} – найменше значення якості даних, q^{max} є максимальним значенням якості даних.

$$QoS_C = \frac{q^{min} - x}{q^{max} - q^{min}}, \quad (2)$$

де QoS_C являє собою значення, отримане шляхом порівняння репутації та вартість виконання, у даному випадку вартість пошуку, x – значення релевантності інформації, які необхідно надати користувачеві, q^{min} – найменше значення релевантності інформації, q^{max} є максимальним значенням релевантності інформації.

$$QoS_A = \frac{x}{100}, \quad (3)$$

де QoS_A являє значення порівняння наявності, а x – значення швидкості надходження інформації, необхідних для порівняння.

$$\text{QoS}_T = T_1 - T_0, \quad (4)$$

де QoS_T являє значення порівняння часу відгуку, T_1 – значення часу відповіді на запит, а T_0 – значення часу, коли був зроблений запит.

Після того, як оцінка всіх даних розрахована і скоригована, обчислюється загальна оцінка веб-служби, щоб представити його як вагу результатів тесту, а не нуль дефектів через ваговий показник якості обслуговування.

$$TCW_{QoS} = 1 - \left(\frac{T_i + C_i + A_i + R_i}{QoS_{attributes}}\right), \quad (5)$$

де TCW являє собою загальний ваговий показник якості обслуговування веб-служби, і, який є менш ніж задовільним.

Таблиця 2.1

Атрибути QoS

QoS attribute	Визначення	Нотації
Час відгуку (T)	Проміжок часу між запитом і відповіддю	Значення часу отримання інформації
Вартість виконання (C)	Вартість, пов'язана з викликом послуги	Вартість послуги в даному випадку вартість пошуку
Наявність (A)	Ймовірність того, що інформація доступна і готова для негайного використання	В даному випадку затримка між API та базою даних
Репутація (R)	Вимірювання надійності служби	Розраховується на основі досвіду користувачів

Також для розроблення методу пошуку були поставлені такі наступні критерії, які були знайдені за допомогою дерева проблем (рис. 2.3) сучасних пошукових сервісів.

1. Вирішення проблеми взаємодії інформаційно-пошукової системи та розробника сайтів: такій проблемі притаманне те, що для того, щоб сайт потрапив до рейтингу сайтів у пошуковій видачі певного пошукового сервісу, потрібно переробити сайт під потреби алгоритму Павука в даній пошуковій системі, тобто зробити велику кількість роботи з аналізу та статистики роботи алгоритму Павука в пошуковій системі та проробити велику кількість роботи з налаштування власного сайту для певної пошукової видачі. Оскільки такий алгоритм часто змінюється, потрібно слідкувати за оновленням даного алгоритму, і це тільки під певну пошукову систему, а їх дуже багато.
2. Адаптація пошуку до потреб користувача.
3. Вирішення проблем неоднорідності джерел інформації.

Організації можуть досягти якості обслуговування, використовуючи певні інструменти і методи, такі як формування трафіку. Для багатьох організацій якості обслуговування включено в угоду про рівень обслуговування SLA з їх постачальником мережевих послуг, щоб гарантувати певний рівень продуктивності.

2.2. Математична модель розроблюваного методу

Перший крок – це створення сервісу, який буде надавати можливість проектувати нову інформаційно-пошукову систему за певною категорією та заповнювати її інформацією. Заповнення інформації відбувається за допомогою системи послідовних об'єктів, що являє собою список. Об'єкти містять інформацію за ключами, за допомогою цих ключів ми зможемо налаштувати роботу з інформацією та збільшити релевантність пошуку.

Для того, щоб налаштувати роботу із інформацією, потрібно налаштувати типи ключів. Всього є декілька типів ключів: це рядок, число, логічний вираз та посилання на інший об'єкт у списку.

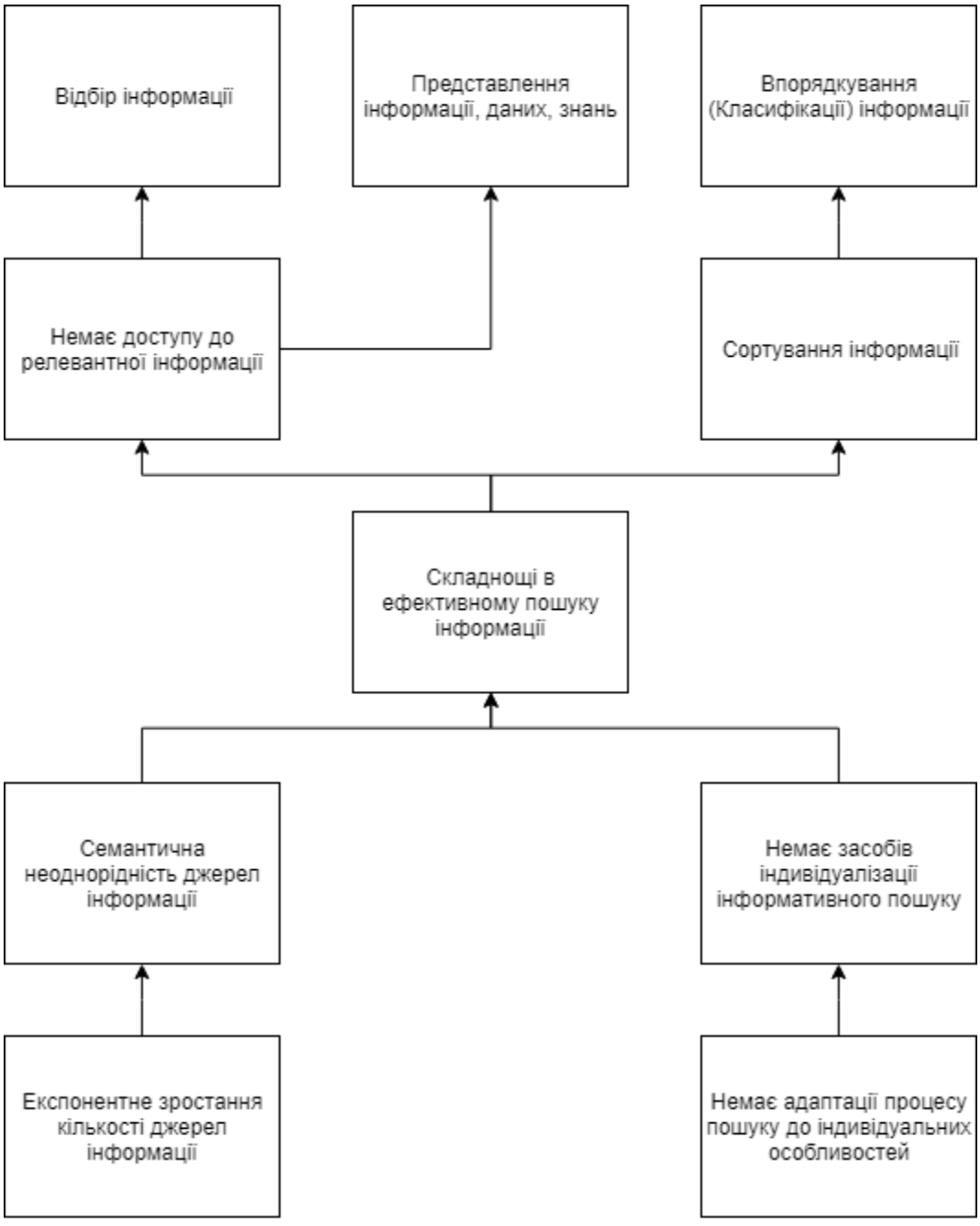


Рис. 2.3. Дерево проблем сучасних пошукових систем

Налаштування роботи з інформацією відбувається двома різними способами.

Перший спосіб – це налаштування за допомогою інструментів графічної обгортки сервісу, а саме вибір ключа інформації, спосіб

зрівняння інформації та вибір головного ключа, за яким буде відбуватися пошук інформації.

Другий спосіб – використання програмного JS коду та розробленої власної бібліотеки, яка буде надавати методи для роботи з інформацією у створеному пошуковому сервісі.

Обов’язковим етапом даної розробки є налаштування безпеки програмного коду, написаного розробником пошукових систем з використанням принципу Керкгоффса [41]. Код відправляється на сервер, де він проходить аналіз на оптимізацію, знаходження CSRF (Cross-Site Request Forgery) та складність алгоритму. Після цього приймається рішення про використання даного коду в інформаційно-пошуковій системі.

Сервіс містить графічні інструменти для оновлення, зберігання та видалення інформації у пошуковій системі.

Для збільшення вартості пошуку кінцевого користувача здійснюється аналіз налаштування пошукового сервісу. Відбувається виведення результатів, у якому вказуються недоліки пошукової системи розробнику даної системи, та генерується документація щодо її використання на основі налаштувань кінцевому користувачеві.

Наступний крок – це алгоритм сортування пошукових систем за категоріями з використанням методу PRF [42]. Суть методу PRF полягає в тому, що ми проводимо пошук в два етапи.

На першому етапі ми для кожного запиту кінцевого користувача обчислюється значення Score – показник релевантності пошукової системи до запиту, на підставі якого здійснюється сортування.

Для розрахунку Score була обрана адаптована модель. Інтерес представляє питання, які складові потрібно додати у формулу розрахунку Score. Під час аналізу ІПС було відібрано наступні складові: збіг слів із запиту в пошуковій системі W_{single}, збіг пар слів із запиту W_{pair}, 100% збіг тексту запиту W_{Phrase}. Крім цього, є складова, що дає пріоритет за наявністю всіх слів запиту в пошуковій системі $W_{AllWords}$.

Наступний крок – проведення k ітерацій для розрахунку Score. Підсумкова формула виглядає наступним чином:

$$Score = \frac{W_{single}+W_{pair}+\frac{1}{k}W_{AllWords}+kW_{Phrase}+W_{PRF}}{k}. \quad (6)$$

Всі ці складові враховуються за допомогою векторів рейтингу під назвою рейтинговий словник конкретного пошукового сервісу. Рейтинговий словник має вигляд:

$$RatingVocabulary = [ScoreWord_0, \dots, ScoreWord_{countWords}] \quad (7)$$

У представленні (7), $ScoreWord$ – це Score певного слова, за допомогою якого вибирається пошуковий сервіс, а $countWords$ – кількість слів у рейтинговому словнику.

В свою чергу, рейтинговий словник у пошуковому сервісі враховується динамічно та при таких критеріях:

1. Якщо у рейтинговому словнику немає жодного слова, то всі слова запиту, при якому був вибраний цей пошуковий сервіс, додаються до рейтингового словника з початковим Score який дорівнює 1.
2. Якщо слова немає в рейтинговому словнику пошукового сервісу, то для того, щоб він потрапив туди, потрібно, щоб виконувалась умова (8).

$$Score_{wordQ} \geq Average_{RV} \quad (8)$$

$Average_{RV}$ виражається наступною формулою:

$$Average_{RV} = \frac{1}{countWords} \sum_{i=0}^{countWords} ScoreWord_i, \quad (9)$$

де $ScoreWord$ та $countWords$ взяти із представлення (7).

$Score_{wordQ}$ – це вектор Score всіх слів, які були в запиті до пошукового сервісу, має представлення:

$$Score_{wordQ} = [ScoreQ_0, ScoreQ_{1,\dots,}ScoreQ_{countQWords}], \quad (10)$$

в свою чергу $ScoreQ_i$ обчислюється за формулою:

$$ScoreQ = \left\lfloor \frac{ScoreQ_{lastmonth}}{Range_{last}} \right\rfloor Range_{last} + \frac{1}{2^4} Range_{quarter} + \frac{1}{2^7} Range_{year}, \quad (11)$$

де $Range$ – це кількість вживання слова в запитах в певний період часу, в даному випадку $Range_{last}$ теперішній місяць, $Range_{quarter}$ останні 4 місяці, $Range_{year}$ останні 12 місяців, а $ScoreQ_{lastmonth}$ – це рейтинг слова за попередній місяць.

За допомогою цих умов слово потрапляє до рейтингового словника пошуково сервісу, але нам потрібно видаляти із словника слова, які могли туди потрапити помилково, слова, які вже втратили свою актуальність у пошуку для даного сервісу, чи слова, які вже не використовуються у пошуковому сервісі. Тому слово залишається в словнику, якщо виконуються наступні умови:

$$\begin{cases} Range_{SWMounth} \geq Average_{RV} \\ Range_{ScoreWord} > 0 \end{cases}, \quad (12)$$

де $Range_{SWMounth}$– це *Score* слова за останій місяць, а $Range_{ScoreWord}$ це *Score* слова за останні 4 місяці не враховуючи теперішній місяць.

Складові W_{single} є вектором W_{vector}, розмір якого дорівнює розміру рейтингового словника із представлення (7). Має вигляд:

$$W_{vector} = [SingleWord_0, SingleWord_{1,\dots,} SingleWord_{countWords}] \quad (13)$$

Будується за такими математичними правилами:

$$\begin{cases} SingleWord \in RatingVocabulary \Rightarrow ScoreWord \\ SingleWord \in W_{vector} \Rightarrow 0 \end{cases}, \quad (14)$$

тоді значення W_{single} мають вигляд:

$$W_{single} = \sum_{i=0}^{countWords} ScoreWord_i * SingleWord_i, \quad (15)$$

Складова W_{pair}, має дещо складніше представлення, оскільки потребує враховування парі слів із запиту. Тому спочатку ми будуємо співвідношення вибраного пошукового слова по іншим словам із запиту.

Наступна складова W_{pair} має вигляд:

$$W_{pair} = \sum_{j=0}^{countQueryWords} \sum_{i=0}^{countWords} SelectWord_j * SingleWord_i \text{ , (16)}$$

де $selectWord$ вибране слово із пошукового запиту, $countQueryWords$ кількість слів у запиту, $SingleWord$ беремо із представлення (13).

Також нам потрібно застосувати метод праймінгу. Цей метод винайдений психолінгвістами. Праймінг – це вплив, що несе за собою більш точне або швидке вирішення завдання щодо цього або подібного впливу, це методичний прийом, в якому подібний вплив є ключовим фактором. Для опису цього явища доречно поняття предналаштування.

За сферою психічних явищ, в яких викликається праймінг, розрізняють когнітивний і емоційний праймінг, за фактом присутності прайму в свідомості – надпороговий і підпороговий, за формою пред'явлення прайму і цільового стимулу – образний, вербальний і комбінований праймінг, за глибиною переробки прайму – сенсорний, перцептивний і семантичний [43].

Для вербального праймінгу виділяють ряд форм праймінгу мовної форми: лексичний, граматичний і синтаксичний. У подальшому будемо використовувати метод праймінгу для покращення результату пошуку нашого пошукового сервісу.

У дослідженнях уваги людини праймінг використовується як інструмент для вирішення ряду наступних наукових проблем.

1. Проблема уваги як відбору, один з аспектів якої – пошук етапу в системі переробки інформації, де припиняється подальший аналіз інформації, що не має відношення до поставленого завдання, і триває глибший аналіз потрібної інформації. Зокрема, праймінг ефекти можуть виявитися корисними для вибору на користь моделей раннього або пізнього відбору.

2. Проблема розподілу ресурсів. Різні види праймінгу, виділення яких засноване на глибині переробки інформації, можуть призводити до різних ефектів при різній завантаженні системи переробки інформації. Тому праймінг-ефекти можуть виявитися корисними при тестуванні пошукових моделей поведінки користувачів ІПС.

В нашому випадку праймінг впливає на наступні когнітивні дії користувача. Наприклад, якщо користувач спочатку знайшов пошуковий сервіс певної категорії, то, скоріш за все, наступна дія буде знайти інформацію у цьому сервісі.

Тому у нас також є список слів, за когнітивними діями, будується аналогічно представленням (7) та має вигляд:

$$W_{prime} = [PrimeWord_0, PrimeWord_{1,\dots,} PrimeWord_{countWo} \quad (17)$$

Потім у складової $W_{prepair}$ ми проходимо по всіх словах та шукаємо слова які спів падуть із словами у представленні та перемножуємо їх на самих себе, тоді ми можемо розрахувати значення $W_{pair,}$ яке має представлення:

$$W_{single} = \sum_{i=0}^{countWords} ScoreWord_i * PrimeWord_i. \quad (18)$$

Складова $W_{AllWorlds}$ має представлення:

$$W_{allWorlds} = \sum_{i=0}^{countWords} ScoreWord_i * SelectWord_i. \quad (19)$$

Аналогічним способом будується складова W_{phrase}, та виглядає:

$$W_{phrase} = \sum_{i=0}^{countWords} W_{allWorlds} * QueryWorlds, \quad (20)$$

де $QueryWorlds$ це групування слів від початку до кінця запиту.

2.3. Висновки до розділу 2

У даному розділі на основі виділених проблем існуючих методів пошуку інформації в мережі Інтернет та запропоновано модифікований метод пошуку інформації на основі адаптації процесу пошуку до

індивідуальних особливостей користувача, наведено математичну модель розробленого методу.

Розглянуті алгоритми пошуку витрачають більш ніж половину ресурсів кластерних систем серверів для підтримки інформації в актуальному стані. Тому у запропонованому методі вирішено відмовитись від сканування веб-сторінок та перенести інформацію у власну інформаційно-пошукову систему.

У розробленій інформаційно-пошуковій системі для покращення реалізації пошукової видачі інформації було використано метод PRF, наведена його математична модель для даної пошукової системи, також застосований метод праймінгу в алгоритмі уточнення пошукової видачі, на підставі якого здійснено покращення видачі релевантної інформації розроблюваного пошукового сервісу.

У запропонованій інформаційно-пошуковій системі не потрібно сканувати веб-сторінки, оскільки система автоматично дізнається про додавання, видалення та оновлення інформації у пошуковому сервісі, що заощаджує ресурси кластерної системи серверів. Система не працює із стандартами WC3, тому не потрібно витрачати багато часу для того, щоб утримати цю систему в актуальному стані. Також даній системі не потрібно індексувати веб-сторінки, а зберігати інформаційні дані у власному форматі, що заощаджує серверний простір на 5%.

Покращенням даного методу є також полегшення роботи з пошуковим сервісом для користувачів, які створили даний сервіс завдяки відмові від налаштування веб-сторінок.

3. РОЗРОБКА СИСТЕМИ ДИНАМІЧНОГО ПРОЕКТУВАННЯ ПОШУКОВОГО СЕРВІСУ ТА ОПИС СТЕКУ ТЕХНОЛОГІЙ

3.1. Вимоги до розроблюваної інформаційно-пошукової системи

При розробці даної системи передбачається, що кінцевий користувач має такі можливості:

1. Зробити запит до пошукової системи для отримання релевантної інформації;
2. Знайти потрібний пошуковий сервіс;
3. Використовувати знайдений пошуковій сервіс для роботи з інформацією;
4. Залишити оцінку пошуковому сервісу після його використання;
5. Залишити зворотний зв'язок для розробника пошукових сервісів, знайденого пошукового сервісу;
6. Залишити зворотний зв'язок для пошукової системи;

Розробник пошукових сервісів має такі можливості, як і кінцевий користувач, він може:

1. Створювати новий пошуковий сервіс;
2. Наповнювати власний пошуковий сервіс релевантною інформацією;
3. У будь-якій час зробити оновлення інформації в пошуковому сервісі;
4. Виконувати налаштування свого пошукового сервісу за власними критеріями;
5. Отримувати зворотній зв'язок, залишений кінцевими користувачами для аналізу пошукового сервісу.

Також розробник пошукового сервісу повинен мати аналітичну інформацію для покращення свого пошукового сервісу, яку він буде отримувати із серверу.

Ця інформація містить такі дані:

1. Кількість запитів до пошукового сервісу;
2. Кількість активних користувачів пошукового сервісу;
3. Кількість користувачів в реальному часі;
4. Найпопулярніші запити до пошукового сервісу;
5. Місце знаходження користувачів;
6. Демографічні дані кінцевих користувачів;
7. Технології, які використовують користувачі для роботи із пошуковим сервісом.

Для покращення збору аналітичної інформації розробник пошукового сервісу матиме можливість налаштувати за допомогою функцій зворотного виклику в мові програмування JavaScript такі виклики:

1. queryCallback – виклик, який буде зроблений після заверешення введення запиту до пошукового сервісу;
2. selectedServiceCallback – виклик, який буде зроблений після того, як кінцевий користувач перейшов до створеного та налаштованого пошуковго сервісу розробника пошукових систем;
3. workInforamtionCallback – виклик, який буде зроблений після того, як кінцевий користувач почав використовувати ключі, які налаштовані розробником пошуковго сервісу, для покращення отримання релевантної інформації, або роботи із інформацією, яка була надана при першому запиті;
4. closeSearchServiceCallback – виклик, який буде зроблений після того, як користувач залишить пошуковий сервіс;
5. switchSearchServiceCallback – виклик, який буде зроблений після того, як користувач змінить пошуковий сервіс на інший;
6. searchServiceExtraditionCallback – виклик, який буде зроблений після того, як користувач побачить даний пошуковий сервіс в пошуковій видачі при першому запиту на пошук сервісів.

Дані виклики отримують такі аргументи як:

1. Id – кінцевого користувача, якій зберігається у форматі Universally Unique Identifier;
2. Request – запит користувача в пошуковому сервісі;
3. Country – код країни, з якої кінцевий користувач зробив запит, даний код зберігається у форматі ISO 3166-1 alpha-3;
4. Language – код мови, використаний кінцевим користувачем для написання запиту в пошуковому сервісі, яка зберігається у форматі ISO 639-2 Code;
5. TimeQuery – час, коли був зроблений запит в пошуковий сервіс, зберігається у форматі ISO 8601;
6. UseKeysService – ключі, які використував кінцевий користувач для для покращення отримання релевантної інформації або роботи із інформацією, яка була надана при першому запиті;
7. countViews – кількість переглядів кінцевим користувачем даного пошукового сервісу.

Сервіс містить графічні інструменти для оновлення, зберігання та видалення інформації у пошуковій системі, за допомогу яких розробник пошукових сервісів зможе працювати с власним пошуковим сервісом.

Налаштування роботи з інформацією відбувається двома різними способами.

Перший спосіб – це налаштування за допомогою інструментів графічної обгортки сервісу, а саме вибір ключа інформації, спосіб зрівняння інформації та вибір головного ключа, за яким буде відбуватися пошук інформації.

Другий спосіб – використання програмного JS коду та розробленої власної бібліотеки, яка буде надавати методи для роботи з інформацією у створеному пошуковому сервісі.

Для збільшення вартості пошуку кінцевого користувача здійснюється аналіз на основі наступних вимог.

8. Пошук групи слів – пошукова система сприймає групу слів в запиті так, ніби між ними стоїть сполучник, як єднальні наприклад, єднальні, «І, ТА, Й», або протиставні, наприклад, «А, АЛЕ, ПРОТЕ», тобто відбувається пошук пошукових сервісів, в яких одночасно зустрічаються обидва слова.
9. Словоформи – у зв'язку з тим, що у лінгвістичних мовах слова змінюються за відмінками, важливою особливістю пошукової системи є пошук популярних словоформ. Пошукова система у більшості випадків дозволяє знаходити різні утворені словоформи, наприклад, останній запит на пошук «популярні програмісти» рівносильний запиту «популярна програма».
10. Роль великих літер – існує загальна умова для більшості пошукових систем, яка полягає в тому, що великі літери на початку слова сприймаються як певна умова, що обмежує пошуку за запитом. Наприклад, за запитом «висота Ейфелевої вежі» будуть знайдені лише ті пошукові сервіси, які містять слова «висота Ейфелевої вежі».
11. Шукати у знайденому – якщо у результаті пошуку було знайдено дуже багато пошукових сервісів, тоді користувач має можливість скоротити даний список, для цього була створена послуга «Шукати у знайденому», яку надають ІПС на панелі керування запитом. Наприклад, запит «Електронна техніка» можна уточнити запитом «серверна».

Система також має можливість генерувати документацію за пошуком під час налаштування пошукового сервісу. Відбувається виведення результатів, у якому вказуються недоліки пошукової системи розробнику даної системи та генерується документація щодо її використання на основі налаштувань кінцевому користувачеві.

Для описання можливостей, що надає розроблювана система, була розроблена діаграма прецедентів (рис. 3.1).

Рис. 3.1. Діаграма прецедентів

Діаграма прецедентів – це граф, що складається з множини акторів, прецедентів (варіантів використання), відношень між акторами та прецедентами.

3.2. Загальна архітектура системи

Архітектура «клієнт-сервер» є одним із архітектурних шаблонів проектування програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти (рис. 3.2):

- серверна частина;
- клієнтська частина;
- прикладний програмний інтерфейс [44].

Перевагами даної архітектури є наступні.

- Відсутність дублювання програмного коду сервера та програмного коду клієнтської частини.
- Так як увесь програмний код виконуються на сервері, то вимоги до характеристик комп'ютерів, які використовує клієнт, знижуються.
- Всі необхідні дані зберігаються на сервері, який захищений набагато краще за більшість пристроїв клієнтів. На сервері простіше організувати контроль повноважень, щоб надавати доступ до даних тільки тим клієнтам, у яких є відповідні права доступу.

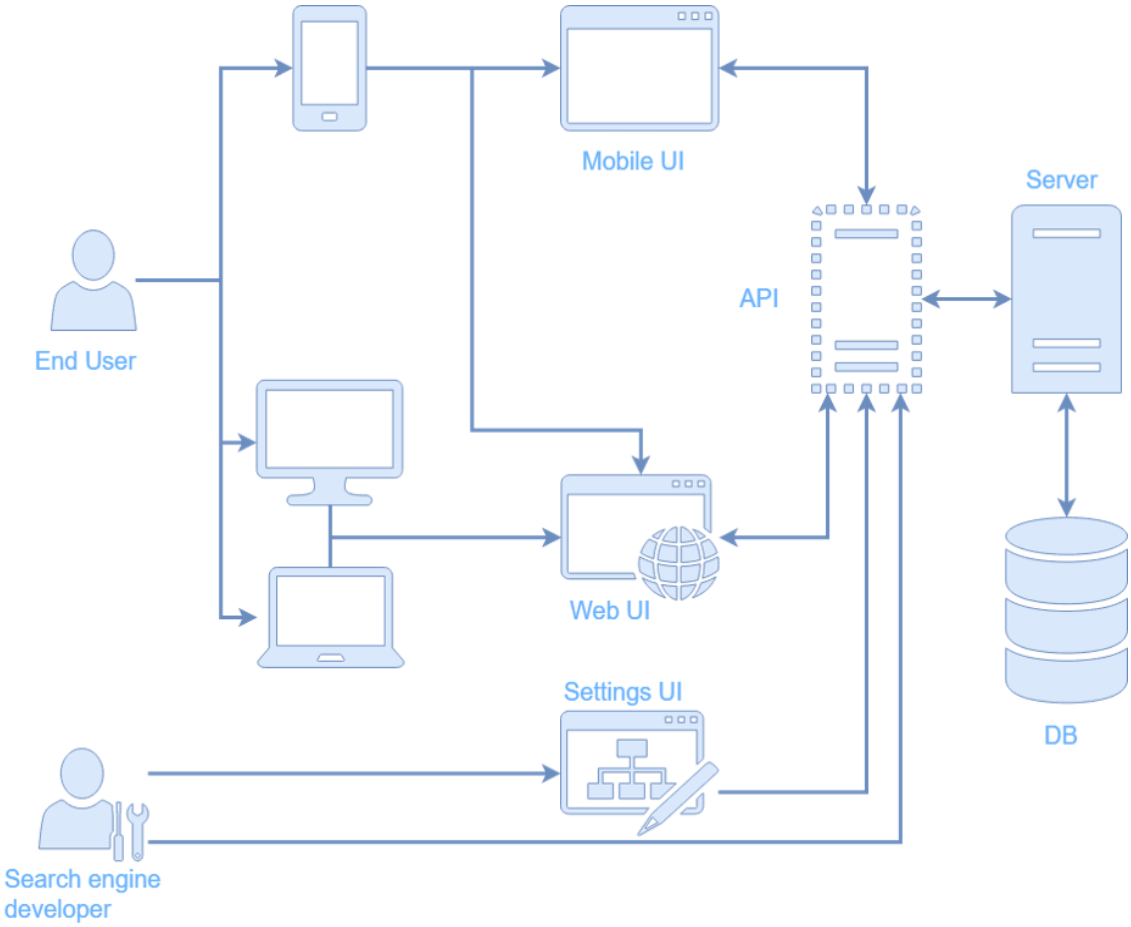


Рис. 3.2. Загальна архітектура програми

Недоліками даного архітектурного рішення є наступні.

- Непрацездатність ключових серверів може зробити непрацездатною всю обчислювальну систему. Непрацездатним

сервером потрібно вважати сервер, який перестав відповідати на запити, виконувати програмний код, продуктивності такого серверу не вистачає на обслуговування клієнтів.

- Підтримка роботи даної системи вимагає кваліфікованого фахівця – системного адміністратора.
- Великий час налаштування та висока вартість обладнання.

Є декілька різновидів архітектури «клієнт-сервер», якщо говорити про багаторівневу архітектуру взаємодії клієнта та сервера, як приклад можна привести будь-яку сучасну СУБД. Суть багаторівневої архітектури полягає у тому, що запит клієнта обробляється відразу декількома серверами. Такий підхід дозволяє значно знизити навантаження на сервер через те, що відбувається розподіл операцій, але в той же час даний підхід не такий надійний, як дворівнева архітектура. Перший рівень можна вважати браузером, за допомогою якого відвідувач заходить на сайт, другий рівень – це програмний код API, а третій рівень – це база даних, в якій функція обробки даних винесена на один або кілька окремих серверів. Це дозволяє розділити реалізацію обробки, зберігання та вигляд даних для більш ефективного використання різних можливостей серверної та клієнтської частини.

Архітектура «клієнт-сервер» не ділить машини на тільки клієнт або тільки сервер, а скоріше дозволяє розподілити навантаження і розділити функціональні можливості між клієнтською частиною і серверною.

Отже, основна ідея цієї архітектури полягає в розділенні мережевого додатку на кілька ключових компонентів, кожен з яких реалізує певний набір сервісів. Компоненти цього додатку можуть працювати на різних комп'ютерах, виконуючи клієнтські та серверні функції. Це підвищує безпеку, надійність і продуктивність мережевих застосунків і мережу, яку використовує система в цілому.

До серверної частини відносяться такі елементи, як Server, який написаний на мові програмування JavaScript та нереляційна база даних

MongoDB. Серверна частини реагує на виклики із прикладного програмного інтерфейсу та вносить нові дані до бази даних, також залежить від запиту прикладного програмного інтерфейсу, дістає і робить валідацію даних із бази даних та передає їх до відповідного запиту із прикладного програмного інтерфейсу. Наприклад, дістати із бази даних всі пошукові сервіси, зробити валідацію головної інформації із відповідного запиту прикладного програмного інтерфейсу та відправити відсортовані дані до цього запиту.

Прикладний програмний інтерфейс – це набір підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Це набір чітко визначених методів для взаємодії різних компонентів у загальній архітектурі, об'єднання серверної та клієнтської частини.

Прикладний програмний інтерфейс надає розробнику пошукових сервісів засоби для швидкої розробки власного пошукового сервісу.

Для того, щоб використовувати прикладний програмний інтерфейс для розробки пошукового сервісу, потрібно знати мову програмування JavaScript та використовувати написану для цього мою бібліотеку для роботи із прикладним програмним інтерфейсом.

Розроблювальна бібліотека не буде надавати можливість розробнику пошукового сервісу перехоплювати канал зв’язку між клієнтською і серверною частинами, в результаті чого ми маємо доступ до всієї потрібної нам інформації, що йде через певний канал зв’язку. Дана атака має назву Man-in-the-middle, мета атаки – не тільки доступ до інформації, а й зміна, крадіжка, надавання відкритого доступу інформації. Гарним прикладом цієї атаки є використання прихованої інформації програми для шахрайства в іграх, інформація про ігровий запит, що породжується на стороні клієнтської частини та відправляється на сервер. На її шляху програма перехоплює запит, змінює інформацію за бажанням зловмисника і відправляє запит до серверу замість тієї інформації, яку відправила

клієнтська частина гри, сервер в свою чергу не розуміє, що запит є зловмисним.

Розроблювана бібліотека буде давати можливість робити ін'єкції програмного коду розробника пошукових сервісів на клієнтську частину кінцевих користувачів, загальний принцип яких – впровадження до інформаційних сервісних систем з програмним кодом, який розробив створювач пошукових систем, в хід передачі даних, де код фактично не заважає роботі додатку, але одночасно виробляє необхідну дію розробнику пошукових сервісів.

Обов'язковим етапом даної розробки є налаштування безпеки програмного коду, написаного розробником пошукових систем з використанням принципу Керкгоффса [45]. Код відправляється на сервер, де він проходить аналіз на оптимізацію, знаходження CSRF (Cross-Site Request Forgery) та складність алгоритму. Після цього приймається рішення про використання даного коду в пошуковій системі.

Логіка веб-додатку розподілена між серверною і клієнтською частиною, зберігання даних здійснюється на сервері, обмін інформацією відбувається по мережі, яка працює за допомогою прикладного програмного інтерфейсу. Одним з переваг такого підходу є той факт, що користувачі не залежать від конкретної операційної системи.

Основною перевагою створення веб-застосунків для підтримки основних функцій браузера є те, що ці функції будуть виконуватися незалежно від встановленої у клієнта операційної системи. Ідея полягає в тому, що замість того, щоб писати різний програмний код для операційних систем, таких як Mac OS, Windows, Linux та інших, додаток створюється лише один раз для потрібної платформи та працює на даній платформі. Також не потрібно піклуватися про різну реалізацію CSS, HTML, DOM, SDOM й інших специфікацій у браузерах, тому що додаток виконується на одному й тому ж двигуні JavaScript, а також спрощує подальшу підтримку програмного коду. Крім того, можливість користувача

налаштовувати багато параметрів браузера (наприклад, розмір шрифту, кольори, відключення підтримки сценаріїв) може перешкоджати коректній роботі застосунку.

До клієнтської частини належать такі елементи загальної архітектури, як мобільні пристрій, комп'ютерні пристрої користувачів, які їх використовують, а саме кінцевий користувач та розробник пошукових сервісів. Насамперед головну частину клієнтської частини складають мобільний інтерфейс, веб-інтерфейс та інтерфейс налаштувань, які використовують користувачі ІПС.

Для більшої інтерактивності і продуктивності був створений підхід до написання програмного коду для веб-застосунків AJAX, і який на теперішній час є стандартом для створення прогресивного веб-застосунку. При використанні технології AJAX програмний код веб-застосунку має можливість відправляти запит до сервера у фоновому режимі і не потрібно перезавантажувати сторінку цілком, а лише довантажувати необхідні дані із сервера, що пришвидшує роботу програмного коду та робить її набагато зручнішою для підтримки. Із вплином часу дана технологія була змінена та вбудована до JavaScript двигунів, які використовуються різними браузерами [46].

Розглянемо декілька шляхів для створення веб-додатків на мобільних пристроях.

- Мобільний додаток або «Прогресивний веб-застосунок» виконується безпосередньо на мобільному пристрої з емулятором веб-браузера та зазвичай не потребує доступу до інтернет з’єднання. Типово такі додатки пишуться мовою JavaScript для Android-пристроїв або на Svelte чи Flatter для iOS. Останням часом, такі програмні каркаси, як React, React-Native, Xamarin дозволяють створювати мобільні застосунки відразу для декілька мобільних операційних систем, використовуючи одну мову програмування, наприклад,

TypeScript, замість звичних мов програмування для мобільних додатків.

- Основною частиною є створення адаптивного веб-дизайну для використання його як звичайного веб-сайту так й для одно-сторінкового мобільного додатку, який повинен бути зручний у використанні на мобільних пристроях з невеликими екранами.
- Прогресивний веб-застосунок поєднує звичайний мобільний додаток із веб-сторінками.
- Гібридні застосунки вбудовують веб-сторінку всередину емулятора на основі двигуна V8, та здійснюють обгортку мобільного додатку. Можуть бути створені за допомогою багатосортних програмних каркасів. Цей підхід дозволяє розробникам використовувати існуючі веб-технології, разом із збереженням певних переваг саме мобільних додатків: застосування апаратного прискорення, offline-операції, доступ до магазину додатків.

3.3. Серверна частина

Для серверної частини була обрана мова програмування JavaScript на платформі nodejs.

Nodejs – це JavaScript оточення, побудоване на JavaScript рушієві V8. Як асинхронне подійне JavaScript оточення, nodejs спроектований для побудови масштабованих мережевих додатків. Для кожного з'єднання викликається функція зворотного виклику, проте, коли з'єднань немає, nodejs зупиняє свою роботу. Це контрастує з більш загальною моделлю, в якій використовуються паралельні OS потоки [47].

Такий підхід є відносно неефективним та дуже важким у використанні програмісту, але користувачі nodejs можуть не турбуватися про блокування процесів, оскільки немає жодних блокувань. Майже жодна

з функцій у nodejs не працює напряму з потоками вводу та виводу, тому процес не блокується ніколи.

Оскільки не відбувається блокування, на nodejs легко розробляти масштабовані програмні системи. В nodejs немає виклик початку циклу подій. Nodejs просто входить в подієвий цикл після запуску виконавчого файлу на виконання. Nodejs виходить з подієвого циклу тоді, коли не залишається зареєстрованих функцій зворотного виклику. Така поведінка схожа на поведінку браузерного JavaScript, де подієвий цикл прихований від користувача.

Продуктивність є основною перевагою nodejs, в той час як інші рішення на стороні сервера створюють потік не менше ніж 2 мегабайти для кожного вхідного з'єднання, і, звичайно, створення потоку відбувається набагато повільніше, ніж розподіл купи пам'яті. Серед інших переваг – орієнтований на події та неблокуючий рушій.

Розробники інформаційно-пошукових системи несуть відповідальність за застосування логіки для вирішення різних проблем. Всі проблеми, з якими стикаються програмісти у клієнтській частині, мають аналоги на серверній частині. Вони виражаються по-різному, але вони в основному одні й ті ж. Nodejs просто висловлює цей факт, об'єднуючи методології дискретним і раціональним чином.

Завдяки тому, що фоновий код написаний таким же чином, застосовуючи ті ж логічні шляхи, він робить перехід набагато більш прозорим, спрощуючи мою роботу та роботу наступних розробників пошукової системи.

Nodejs для роботи із іншими програмними модулями використовує пакетний менеджер npm (node package manager). За допомогою npm можна встановлювати пакети локально або глобально.

У локальному режимі пакети встановлюються в каталог ./node_modules батьківського каталогу. Власником каталогу є поточний користувач. Глобальні пакети встановлюються у каталог, власником якого

є root в операційній системі. Також при створенні нового проекту та додавання до нього інших модулів створюється файл package.json, (рис. 3.3).

```
{
  "name": "server",
  "version": "0.0.1a",
  "description": "kpi work by Kolomiiets Ivan",
  "main": "-",
  "repository": "git@gitlab.com:klolo966/apidiplom.git",
  "author": "kolomiiets",
  "license": "MIT",
  "private": true,
  "dependencies": {
    "@babel/cli": "^7.1.2",
    "@babel/core": "^7.1.2",
    "@babel/node": "^7.0.0",
    "@babel/preset-env": "^7.1.0",
    "babel-preset-minify": "^0.5.0",
    "express": "^4.16.4",
    "global": "^4.3.2",
    "lodash": "^4.17.11",
    "luxon": "^1.5.0",
    "nodemon": "^1.18.5"
  },
  "scripts": {
    "start": "nodemon --exec babel-node src/index.js",
    "build": "babel src/index.js --out-file dist/server.js"
  },
  "devDependencies": {}
}

```

Рис. 3.3. Структура файлу package.json серверної частини

У даній структурі ми бачимо такі поля:

1. name – назва проекту;
2. version – версія проекту;
3. description – короткий опис проекту;
4. main – головний виконавчий файл, у даному випадку він не заданий, тому що ми використовуємо модуль nodemon;
5. repository – гіперпосилання на репозиторій, де розміщені вихідні файли проекту;
6. autor – автор проекту, або головний лідер проекту;
7. license – ліцензія проекту, в даному випадку у нас є ліцензія MIT(Massachusetts Institute of Technology), розроблена Массачусетським технологічним інститутом для розповсюдження вільного програмного забезпечення. Суть її

полягає в тому, що дійсне програмне забезпечення надається без гарантій, непрямих або прямих, включаючи гарантії комерційної вигоди, відповідності їх конкретному призначенню й відсутності порушення прав. У жодному разі розробники, власники авторських прав або автори не несуть відповідальність за судовими позовами щодо нанесених збитків або будь-яких претензій, чи внаслідок дій договору, суспільного правопорушення або інших, що виникають у зв'язку з експлуатацією програмного забезпечення.

8. private – це поле, що відповідає за те, щоб даний проект був приватним та не повинен потрапляти до глобальних модулів у пакетному менеджері npm.
9. dependencies – важливе поле у файлі package.json, в цьому полі описуються додаткові модулі, які використовуються у серверній частині, це головний модуль babel та його допоміжні модулі, такі як cli, що дає можливість використовувати можливості babel у терміналі, core модуль, який виконує головний принцип babel, це інструмент, який допомагає писати код в останній версії JavaScript. Якщо підтримувані середовища не підтримують деякі функції спочатку, babel допоможе скомпілювати ці функції до підтримуваної версії, незалежно від платформи та операційної системи проекту. Babel node використовується для інтеграції bable core в середовище та екосистему nodejs. Babrl preset event додаткові налаштування для виконавчого середовища bable та трансполяції коду. Babel preset minify запускається після виконаня трансполяції babel та робить мініфікацію коду. Мініфікація – процес, спрямований на зменшення розміру вихідного коду шляхом видалення непотрібних символів без зміни його функціональності. Мініфікація особливо корисна

для програм, написаних на інтерпретованих мовах, тому що вона зменшує обсяг даних, які повинні бути оброблені. Express – швидкий, гнучкий, мінімалістичний веб-фреймворк для додатків nodejs, надає великий набір функцій для мобільних і веб-додатків. Маючи в своєму розпорядженні багато службових методів HTTP і проміжних оброблювачів, створити надійний API можна швидко і легко. Express надає тонкий шар фундаментальних функцій веб-додатків, які не заважають працювати з функціями nodejs. Наступний модуль lodash – сучасна бібліотека JavaScript, що забезпечує модульність, продуктивність та додаткові можливості. Lodash спрощує javascript, позбавляючи від клопоту роботи з масивами, числами, об'єктами, рядками. Модульні методи lodash відмінно підходять для: ітерації масивів, об'єктів і рядків, маніпулювання і тестування значень, створення складених функцій. Головною проблемою в javascrip є те, що він дуже погано працює з датами, тому для вирішення даної проблеми було вирішено додати новий модуль під назвою luxon – обгортку для дат і часу в javascript. Luxon підтримує такі можливості, як типи DateTime, Duration і Interval, незмінний, ланцюговий, однозначний API, розбір і форматування для загальних і призначених для користувача форматів, рідний часовий пояс і підтримка Intl без локалізації або tz файлів. Головний модуль у проекті nodemon – інструмент, який допомагає розробляти програми на основі node.js, автоматично перезапускати додаток вузла, коли виявляються зміни файлу в каталозі. Nodemon не вимагає яких-небудь додаткових змін в вашому коді або методі розробки. Оболонка для заміни nodejs, призначена для того, щоб використовувати nodemon як заміну nodejs в командному рядку при виконанні програмного коду.

10. scripts – у даному полі описуються команди для запуску проекту, в нашому випадку це start запуск модуля nodemon із вихідним файлом index.js, та початок спостереження за змінами в проекті. Команда build робить зборку проекту в єдиний bundle файл для подальшого розповсюдження. Bundle використовує топологічне сортування, щоб визначити порядок, у якому відбувається об'єднання, тому не потрібно турбуватися про те, щоб щось було втрачено. Однак в результаті цього bundle не підтримує циклічні залежності. Виконання роботи являє собою один єдиний програмний код JavaScript, який можна записати у вихідний файл.
11. devDependencies – це модулі, які не попадуть у проект після його збору, загалом тут використовуються модулі для тестування проекту.

До серверної частини також входить база даних. Існують дві реалізації баз даних – нереляційні та реляційні. Відмінності між ними є в тому, як бази даних були спроектовані, які типи або види даних вони підтримують, як працюють та зберігають інформацію. Реляційні бази даних зберігають в собі структуровані дані, які зазвичай мають представлення об'єктів реального світу.

Відомості про користувача, або про вміст пошукової системи, згруповані в модуля даних, формат яких був заданий на етапі проектування архітектури бази даних. Нереляційні бази даних влаштовані інакше. Наприклад, такі бази зберігають інформацію у вигляді довільних ієрархічних структур даних, наприклад, об'єкти з довільним набором атрибутів. Те, що в реляційної базі даних було розбито на кілька взаємопов'язаних модулів або об’єктів, в нереляційних баз даних може зберігатися у вигляді однієї сутності, наприклад JSON файлу.

Модель документа MongoDB проста в освоєнні і використанні для розробників, але при цьому надає всі можливості, необхідні для

задоволення найскладніших вимог в будь-якому масштабі. Тому було обрано нереляційну базу даних, а саме MogoDB.

MongoDB – це база даних документів з масштабованістю і гнучкістю, яка підходить для запитів з індексацією потрібних даних. MongoDB зберігає дані в гнучких, схожих на JSON документах, що означає, що поля можуть варіюватися від документа до документа, а структура даних може змінюватися, з часом модель документа відображається як на об'єкти в коді програми, що спрощує роботу з даними. Спеціальні запити, індексація та агрегація в реальному часі надають потужні способи доступу і аналізу даних. MongoDB є розподіленою базою даних за своєю суттю, тому висока доступність, горизонтальне масштабування і географічний розподіл є вбудованими і простими у використанні [48].

Серверна частина виконує такі операції як: прийняти запит із прикладного програмного інтерфейсу, зробити валідацію та записати дані в базу даних, дістати дані із бази даних, зробити валідацію та відправити їх до запиту із прикладного програмного інтерфейсу (рис. 3.4).

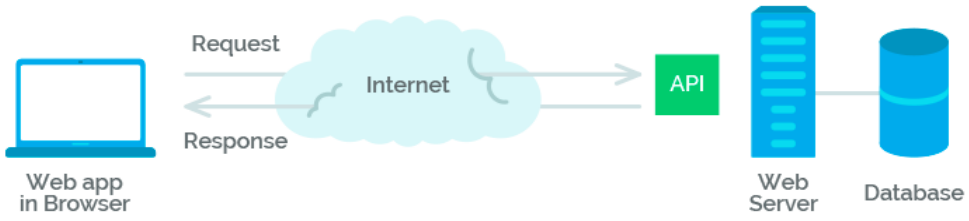


Рис. 3.4. Схема роботи серверної частини

Для роботи серверу із базою даних MongoDB був обраний модуль Mongoose. Mongoose надає просте рішення на основі схем для моделювання даних застосування. Він включає в себе вбудоване приведення типів, перевірку, побудова запитів, функції зворотного виклику бізнес-логіки і багато іншого.

Моделі – це конструктори, складені з схеми визначень. Примірник моделі називається документом. Моделі відповідають за створення і читання документів з ї бази даних MongoDB [49].

Для захисту даних та коректного спілкування із користувачами використовуються система OAuth токенів. OAuth – відкритий протокол авторизації, який дозволяє надати третій стороні обмежений доступ до захищених ресурсів користувача без необхідності передавати їй логін і пароль.

Результатом авторизації є access token, UUID-ключ, згенерований набір символів, який надає доступ до захищених інформаційних ресурсів. Звернення до інформаційних ресурсів відбувається за допомогою HTTPS-протоколу із зазначенням в головних заголовках або в якості одного з параметрів, який починається із «Bearer» та використовує access token. Це є найскладніший варіант авторизації у веб-застосунках, але він дозволяє системі однозначно працювати із додатками, які звертаються за авторизацією до серверу, це відбувається при комунікації між серверами на першому кроці [50].

У нашому випадку ми використовуємо API, і наприклад, коли хтось просить нас поділитися своїм пошуковим сервісом та інформацією у ньому. В даному випадку API є постачальником послуг: в ньому є дані для входу та дані пошукового сервісу. Веб-додаток є споживачем, оскільки користувач хоче використовувати веб-додаток для роботи із інформацією. При наданні цьому веб-додатку доступу до своєї інформації у пошуковому сервісі за допомогою OAuth-сервісу він автоматично у фоновому режимі відсилає веб-додатку access token, за допомогою якого він зможе керувати інформацією пошукового сервісу.

API представляє собою інтерфейс прикладного програмування (рис. 3.5).

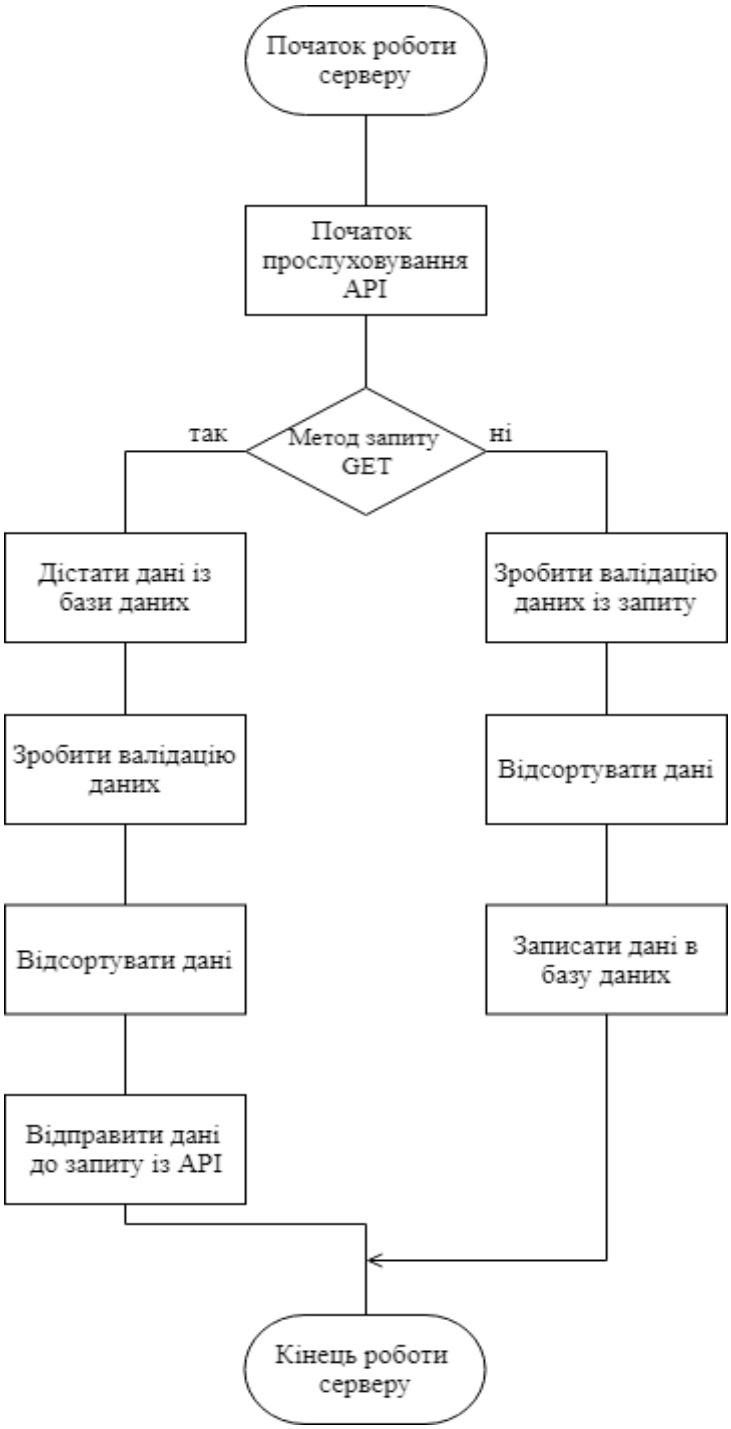


Рис. 3.5. Алгоритм роботи серверної частини

Це набір правил, які дозволяють програмам спілкуватися одна з однією, в нашому випадку ми використовуємо HTTP протокол. За програмного коду ми будемо посилати запити до API. Складається з чотирьох складових:

- Метод – це тип запиту, який відправляється на сервер. Можна обирати один з п'яти типів які наведені в табл. 3.1.

- Заголовки – використовуються для надання інформації як клієнтові, так і серверові. Можуть використовуватися для багатьох цілей, таких як аутентифікація і надання інформації про вміст тіла.
- Дані – містять інформацію для відправлення на сервер. Ця опція використовується тільки з POST, PUT, PATCH або DELETE запитів.
- Кінцева точка – це URL, на який робиться запит. Опис головних реалізованих URL наведені в табл. 3.2.

Таблиця 3.1

Типи методів

Назва типу метода	Значення запиту
GET	Цей запит використовується для отримання ресурсу з сервера. Якщо виконувати запит GET, сервер шукає запитані дані і відправляє їх користувачу. Іншими словами, запит GET виконує операцію READ. Це метод запиту за замовчуванням.
POST	Цей запит використовується для створення нового ресурсу на сервері. Якщо виконувати запит POST, сервер створює новий запис в базі даних і повідомляє вам, чи було створення успішним. Іншими словами, запит POST виконує операцію CREATE.
PUT, PATCH	Ці два запити використовуються для поновлення ресурсу на сервері. Якщо виконувати запит PUT або PATCH, сервер оновлює запис в базі даних і повідомляє, чи було оновлення успішним. Іншими словами, запит PUT або PATCH виконує операцію UPDATE.

Продовження табл. 3.1

Назва типу метода	Значення запиту
DELETE	Цей запит використовується для видалення ресурсу з сервера. Якщо виконувати запит DELETE, сервер видаляє запис в базі даних і повідомляє, чи було видалення успішним. Іншими словами, запит DELETE виконує операцію DELETE.

Таблиця 3.2

Опис реалізацій URL в API

URL	Доступні Методи	Опис
GuestsToken	GET	Створює новий токен для користувача, за допомогою якого він буде використовувати всі інші запити. Даний запит створює початок сесії між користувачем та API, вводиться для того, щоб запобігти спам від зловмисника.
SearchServices	GET, POST, DELETE, PUT, PATCH	Створює, оновлює, видаляє пошукові сервіси у користувача, для того, щоб користуватися даним URL, потрібно мати ClientToken, який видається після того, як користувач зареєструвався або авторизувався у системі.
URL	Доступні Методи	Опис

Продовження табл. 3.2

Auth	GET, POST	За допомогою цього URL ми робимо авторизацію або реєстрацію. Авторизація робиться у два кроки, перший крок – за допомогою метода GET робимо запит на API, який повертає або знайденого користувача та його ClientToken, або статус помилки 404. Якщо ми отримали статус помилки, 404 потрібно за допомогою метода POST зареєструвати нового користувача та видати йому ClientToken.

Коли ми відправили запит до API, він за допомогою HTTP посилає нам статус код, елемент статус коду у відповіді сервера являє собою тризначне ціле число, де перша цифра визначає клас відповіді, а дві останні цифри не мають ніякого значення категоризації. Для першої цифри є 5 значень, а саме:

1. Інформаційні – клас кодів стану вказує попередню відповідь, починаються із одиниці.
2. Успішні операції – клас статус кодів вказує на те, що запит був отриманий та успішно оброблений, починаються із двійки.
3. Перенаправлення – вказує, що клієнт має виконати додаткові дії для завершення виконання запиту, починаються із трійки.
4. Клієнтська помилка – клас кодів, який вказує, що клієнтська частина робить неправильні запити, або на клієнтській частині сталося помилка, починаються із четвірки.

5. Серверна помилка – клас кодів, який вказує, що на сервері або в базі даних сталася помилка і запит не може бути успішно оброблений.

Протокол HTTP використовується для зв'язку між системами, які не поділяють нічого, крім розуміння протоколу. Це дозволяє будь-якій мові програмування, яка розуміє HTTP протокол, користуватися API та за допомогою нього використовувати пошукову систему для створення, оновлення, видання свого пошукового сервісу без використання графічного інтерфейсу.

Кожна схема в Mongoose відображається в колекцію MongoDB і визначає форму документів в цій колекції. Кожен ключ в нашому коді визначає властивість у наших документах, яка буде приведено до пов'язаних з нею схемами типів. Наприклад, ми визначили властивість, title яка буде приведена до типу рядка у даній схемі, і властивість, date, яка буде приведена до типу дати, яка наведена в модулі luxon і зберігається у форматі ISO 8601. Ключам також можуть бути призначені вкладені схеми, що містять додаткові визначення ключів (рис 3.6).

```
import {mongoose} from "mongoose";
const Schema = mongoose.Schema;

const searchService = new Schema({
  title: String,
  author: String,
  body: String,
  comments: [{ body: String, date: Date }],
  date: { type: Date, default: Date.now },
  hidden: Boolean,
  meta: {
    votes: Number,
    favs: Number
  }
});
```

Рис. 3.6. Приклад коду для схеми бази даних

Розглянемо наступні переваги OAuth-авторизації.

1. При використанні OAuth-авторизації користувач не передає свій логін і пароль до захищених ресурсів безпосередньо в додаток.
2. У користувача більше підстав довіряти додатку, оскільки користувач може бути впевнений, що несанкціонований доступ до його особистих даних неможливий. Не володіючи логіном і паролем користувача, додаток зможе виконувати тільки ті дії з даними, які дозволив користувач, і ніякі інші.
3. При розробленні програмного забезпечення не потрібно піклуватися про забезпечення конфіденційності логіна і пароля користувача. Логін і пароль не передаються з додатком, а отже, вони не можуть потрапити в руки зловмисників.
4. Якщо користувач змінює пароль, то додаток більше не може отримати доступ до захищених ресурсів.
5. Єдиний спосіб заборонити додатком доступ до захищених ресурсів – змінити пароль. Це одночасно заборонить доступ до ресурсів і іншим додаткам, які раніше його мали.
6. Сервіси, що зберігають захищені ресурси, які можуть надати прикладний програмний інтерфейс для доступу до них, можуть використовувати федеративні механізми аутентифікації, що дозволяє користувачам не мати пароля до їх акаунтів. Це робить неможливим для цих користувачів використання додатків, на якій знаходиться доступ до захищених ресурсів через цей прикладний програмний інтерфейс.

3.4. Висновки до розділу 3

У даному розділі описаний процес розроблення програмного забезпечення для створення інформаційно-пошукової системи, яке буде надавати можливість проектувати нові пошукові сервіси за певною категорією та заповнювати їх інформацією.

Описані вимоги користувача, функціональні вимоги та можливості розробленої системи, описано обґрунтування вибору стеку технологій.

Проаналізовано сучасні мови програмування, середовища розробки та бібліотеки. Для реалізації було обрано найбільш пристосовані для поставленої задачі засоби.

Для проектування клієнтської частини обрано такі технології як Polyfill для підтримки програмного коду на різних браузерах babel/core, lingui/cli для управління програмною утилітою за допомогою терміналу, lingui/loader, lingui/macro, lingui/react, бібліотека luxon, mobx-devtools та mobx-devtools-mst, набір утиліт mobx-react для інтеграції react станів в mobx, бібліотека react.

Для проектування клієнтської частини обрано технології babel-cli, babel-core, babel-loader, babel-preset-env для підтримки програмного коду на різних операційних системах, бібліотека express для реалізації API та роутерів, lowdb для тестування програмного забезпечення та емуляції бази даних, nodemon для швидкого перезапуску серверу за необхідністю, генератор унікальних ідентифікаторів uuid для списків.

Для зберігання даних пошуково-інформаційної системи обрано нереляційну базу даних MongoDB та рекомендовані в документації розробника MongoDB інструменти Mongod/cli та Mongoose.

Наведено архітектуру, на основі якої виконувалось розроблення програмного забезпечення, інтерфейс користувача.

Описані основні класи розробленого програмного забезпечення.

4. ОЦІНКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОГО МЕТОДУ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ

4.1. Результати ефективності запропонованого методу

Оцінка роботи пошукової системи – це процес прийняття судження про важливість знайденої інформації та якість роботи пошукового двигуна. Після аналізу інформаційно-пошукових систем оцінка рейтингів пошукових систем не встигала за просуванням їх розвитку. Такі системи функціонують по-різному в залежності від інтерфейсу, функцій, навантаження в мережі та методів ранжування. Нелегко оцінити їх на єдиній основі. Існує багато стратегій для оцінки пошукових двигунів, такі, як автоматична оцінка або оцінка користувача на основі судження. Оскільки оцінка користувача пошукової системи має велику похибку, тому ми будемо використовувати автоматичну оцінку на основі критеріїв, які запропонували International Data Group, а саме:

- Базова технологія.
- Масштабованість.
- Провайдери.
- Індексування.
- Релевантність пошуку.
- Безпека інформації.
- Навантаження на ресурси системи [51].

На кожний критерій є декілька питань, і якщо відповідь на питання позитивна, інформаційно-пошукова система отримає бал.

Базова технологія у розробленій інформаційно-пошуковій системі написана на другій в світі популярній згідно статистики Stackoverflow (рис. 4.1) мові програмування JavaScript, як клієнтська частина так і серверна, це забезпечує розуміння програмного коду та їх зв’язування на різних

частинах, також програмний код є відкритим що сприяє вдосконаленню та розвитку пошукової системи, оскільки над проектами з відкритим кодом працюють програмісти, яким подобається те, що вони роблять.

Масштабованість повинна відповісти на питання, чи можливо масштабувати програмний код, оскільки весь написаний програмний код виконано згідно сучасних стандартів, таких як JSG, WC3, GJSG, ISO, SJS, тому він легко масштабується. Після великого зростання інформації системі буде потрібно негайно виділити необхідні їй обчислювальні ресурси.

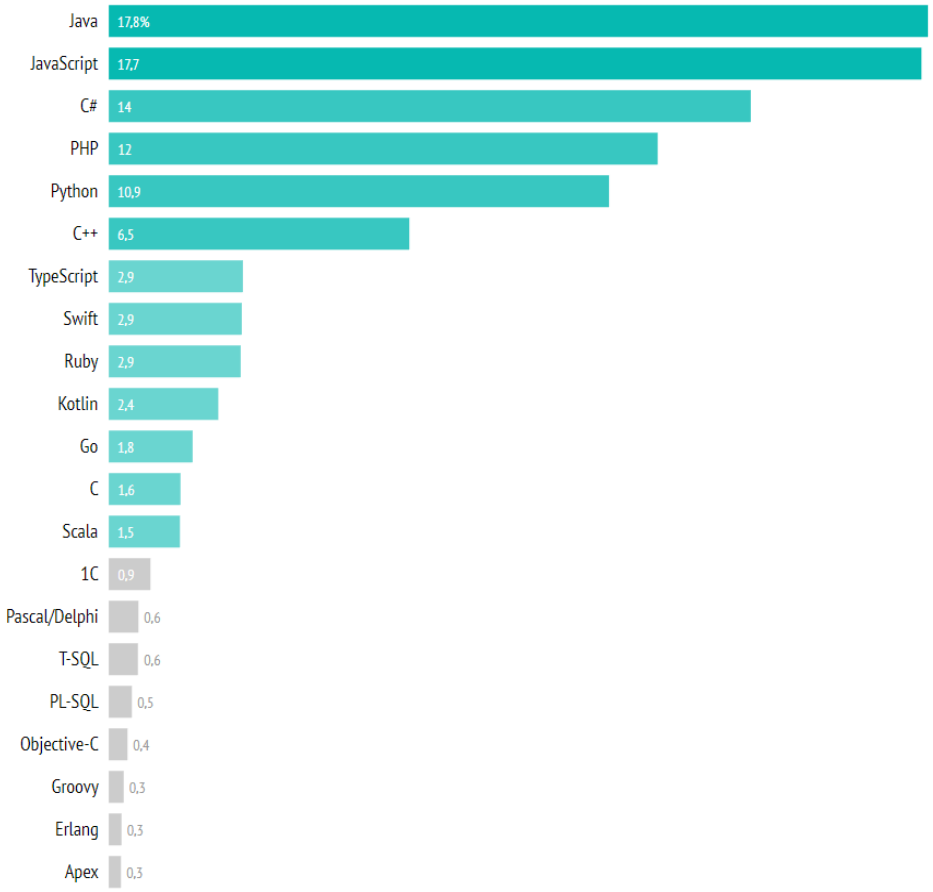


Рис. 4.1. Популярні мови програмування згідно статистики Stackoverflow

Розроблений програмний продукт не підтримує кластерну систему. В розробленій ІПС ми можемо напряму працювати із стороннім програмним кодом. Сканування ІПС має важливе значення для індексації

контенту, в запропонованій ІПС надається перевага не скануванню ресурсів, а зберіганню їх у базі даних з самого початку їх створення. Деякі аспекти індексування, такі як швидкість індексації та затримку оновлення індексації запропонованої ІПС та інших ІПС представлені у таблиці 4.1.

Таблиця 4.1

Порівняння аспектів індексування пошукових систем

Назва пошукової системи	Швидкість індексації в годинах	Оновлення індексації в годинах
Запропонований метод	Від 0,4	Від 0,1
Google	Від 18	Від 0.3 до 48
Yandex	Від 22	Від 0.1 до 24
Baidu	Від 4	Від 1 до 24

Дані з табл. 4.1 для пошукових систем Google, Yandex, Baidu взяті із статистки International Data Group [52]. Ми бачимо, що запропонований метод завдяки відмові від сканування веб-сторінок перемагає у швидкості оновлення та індексації інформації, затримка виника лише через те, що системі потрібно просканувати нову інформацію та розсортувати її у базі даних та перевірити безпеку стороннього програмного коду за необхідністю.

Ранжування за релевантністю – це процес сортування результатів інформації, які з найбільшою ймовірністю належать до запиту користувача, з'являться в верхній частині, в нашому випадку ми використовуємо PRF метод. Чим краще пошукова система розуміє наміри користувача, тим вищу релевантність пошуку можна досягти за допомогою пошукової системи. Тому в запропонованому методі був обраний підхід пошуку інформації за допомогою методу праймінгу.

До даного пункту відноситься не лише безпека зберігання інформації, а й безпека користувача, який може отримати деструктивну інформацію. У запропонованому методі деструктивна інформація знаходиться ще на етапі створення пошукової системи, якщо інформація все одно пройшла перевірку та потрапила в пошукову систему, то на цей

випадок користувач може повідомити систему за допомогою відповідного сервісу.

Навантаження на ресурси системи та перевірка виконується за такими критеріями при різній кількості 100 запитів в секунду:

- Надання інформації
- Швидкість відповіді із серверу
- Швидкість ранжування

Для покращення результатів перевірка робиться в локальній мережі, щоб запобігти похибці на час з'єднання. Результати перевірки наведені в таблиці 4.2.

Таблиця 4.2

Результати тестування навантаження системи

Запропонований метод	Надання інформації в мс	Швидкість відповіді із серверу в мс	Швидкість ранжування в мс	Загалом в мс
Запропонований метод	448	710	2683	3 841
Google	463	200	3372	4 035
Yandex	378	250	3183	3 811
Baidu	446	350	3781	4 577

Дані з табл. 4.2 для пошукових систем Google, Yandex, Baidu взяті із статистки International Data Group. Із табл. 4.2 ми бачимо, що представлений метод виграв під час ранжування інформації та програв у наданні інформації та швидкості відповіді із серверу, в загалом є покращення на 4,8% із Google та на 16,08% зі Baidu. Також була зроблена перевірка на витрати дискового простору для бази даних які зберігають корисну інформацію (рис. 4.2).

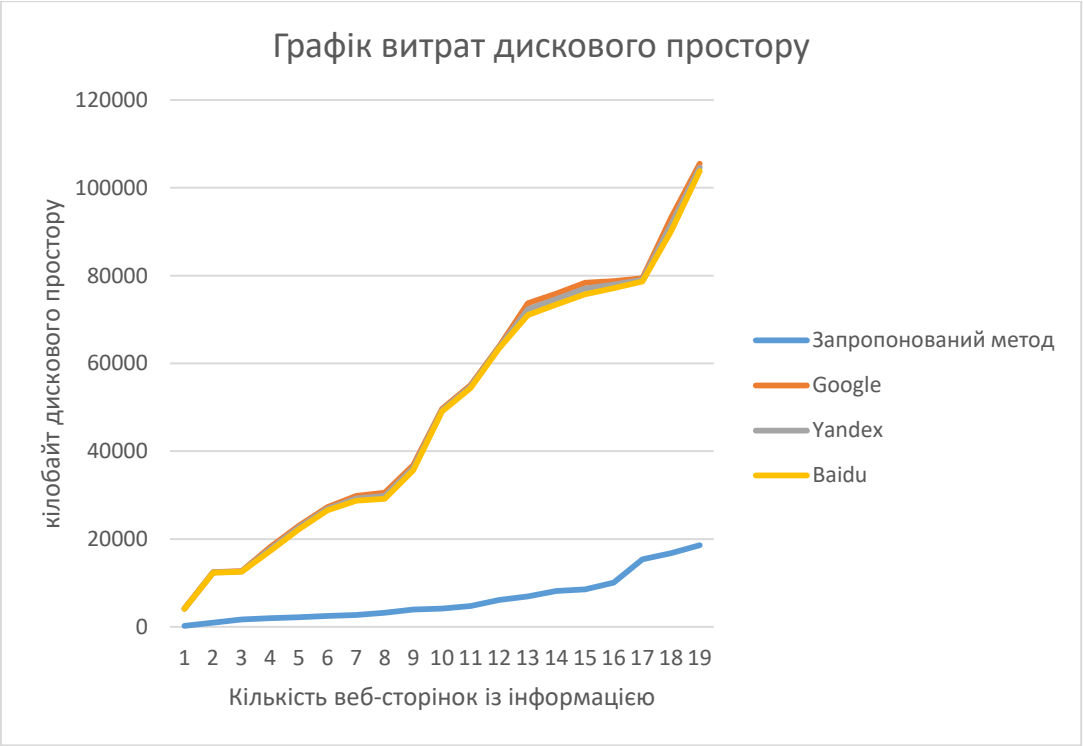


Рис. 4.2. Графік витрат дискового простору

На графіку видно, що всі сучасні інформаційно-пошукові системи використовують індексацію веб-сторінок, тому їм потрібно не лише зберігати корисну інформацію, а ще і розмітку веб-сторінки. З моменту створення CSS3 є можливість також робити маніпуляції з інформацією.

4.2. Тестування клієнтської та серверної частини

Клієнтська частина, а саме програмний код веб-додатку, був протестований за допомогою функціонального тестування, яке є процесом перевірки забезпечення якості програмного забезпечення на основі принципу чорної скриньки. Функції перевіряються шляхом подачі на них вхідних даних і перевірки вихідних даних, а внутрішня структура програми не враховується. Для реалізації функціонального тестування обраний фреймворк JEST. За допомогою даного фреймворку виконано покриття програмного коду (рис. 4.3).

Рис. 4.3. Результати виконання тестів

Для серверної частини також було обраний метод функціонального тестування, але для того, щоб протестувати REST API, була обрана утиліта Test Rest Client (рис. 4.4).

Рис. 4.4. Результати виконання тестів

4.3. Висновки до розділу 4

Завдяки застосуванню методу PRF збільшено швидкість виконання ранжування пошукової видачі на 10.2% порівняно із існуючими алгоритмами пошуку інформації, які використовуються в сучасних інформаційно-пошукових системах.

Для побудови інформаційно-пошукової системи застосовано метод праймінгу в алгоритмі уточнення результатів пошукової видачі, завдяки чому здійснено покращення видачі релевантної інформації пошукового сервісу.

Застосовано новий підхід для роботи із інформацією шляхом відмови від індексування веб-сторінок, завдяки чому відбувається зменшення витрат дискового простору серверу на 17.61% порівняно з витратами дискового простору відомих інформаційно-пошукових систем.

ВИСНОВКИ

У даній магістерській дисертації виділено недоліки сучасних інформаційно-пошукових систем, які використовуються для пошуку та роботи з інформацією у мережі Інтернет.

У першому розділі здійснено аналіз методів пошуку інформації в мережі Інтернет сучасних інформаційно-пошукових систем, зокрема, методів пошуку системи Google, яка в середньому обробляє понад 40 тисяч пошукових запитів за секунду. Виділено переваги та недоліки пошукових систем Yandex та Baidu, які мають широкий спектр різноманітних послуг, включаючи бібліотеку знань, систему обміну файлами та власну платформу соціальних медіа даних.

У другому розділі на основі виділених недоліків існуючих методів пошуку інформації запропоновано та описано новий метод пошуку інформації на основі зведення пошукового запиту до індивідуальних особливостей користувача. Запропонований метод дозволяє збільшити вартість пошуку, замінивши алгоритм ранжування алгоритмом сортування в пошукових системах за категоріями. Інформація заповнюється системою послідовних об'єктів, яка представляє собою список. Об'єкти містять ключову інформацію, за допомогою цих ключів ми можемо налаштувати роботу з інформацією і підвищити актуальність пошуку. Аналіз конфігурації служби пошуку виконується для збільшення вартості пошуку для кінцевого користувача. Система сортування аналізує запит кінцевого користувача і надає знайдену інформацію, відсортовану за релевантністю.

У третьому розділі описано технології розроблення програмного забезпечення для створення пошуково-інформаційної системи, яка буде надавати можливість проектувати нові пошукові сервіси за певною категорією та заповнювати її інформацією. Описані вимоги користувача, функціональні вимоги та можливості розробленої системи, описано обґрунтування вибору стеку технологій для розроблення програмного

забезпечення. Наведено архітектуру, на основі якої виконувалось розроблення програмного забезпечення, інтерфейс користувача, описані основні класи розробленого програмного забезпечення. Проаналізовано сучасні мови програмування, середовища розробки та бібліотеки для проектування ІПС.

У четвертому розділі описано результати експериментів та порівнянь між розробленим та існуючими алгоритмами пошуку інформації в мережі Інтернет. Описано алгоритми та програмні інструменти для тестування та перевірки ефективності розробленої інформаційно-пошукової системи. Розроблений метод забезпечує прискорення швидкості виконання ранжування пошукової видачі на 10.2% порівняно з сучасними алгоритмами пошуку відомих інформаційно-пошукових систем.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Zong Woo Geem. Recent Advances in Harmony Search Algorithm// Studies in Computation Intelligence. – 2016. – P. 51– 75.
2. Schütze, Hinrich, Christopher D. Manning, Raghavan, Prabhakar: Introduction to information retrieval, Prabhakar// Cambridge, UK: Cambridge University Press. – 2008. – P. 121– 125.
3. Автоматизовані інформаційно-пошукові мови. [Електронний ресурс]. — Режим доступу: http://ubooks.com.ua/books/00092/inx13.php. — Дата доступу: Травень 2019. — Назва з екрана.
4. Sergey Brin, Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine// Computer Science Department, Stanford University, Stanfor. – 1998. – P. 45– 48.
5. Google PageRank. [Електронний ресурс]. — Режим доступу: https://ahrefs.com/blog/google-pagerank/. — Дата доступу: Травень 2019. — Назва з екрана.
6. Benjamin M. Schmidt & Matthew M. Chingos: Ranking Doctoral Programs by Placement A New Method// Cambridge, UK: Cambridge University Press. – 2007. – P. 523–529.
7. Matthew Richardson, Pedro Domingos: The Intelligent Surfer:Probabilistic Combination of Link and Content Information in PageRank// The Politics of Search. – 2001. – P. 364–374.
8. Bradley C. Love, Steven A. Sloman: Mutability and the determinants of conceptual transformability// Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society. – 2001. – P. 654–659.
9. Brin, S., Page, L.: The anatomy of Web search engine// Computer Networks and ISDN Systems. – 2001. – P. 107–117.
10. Page, Larry: PageRank: Bringing Order to the Web// Stanford Digital Library Project. – 2002. – P. 47–52.

11. Taher Haveliwala & Sepandar Kamvar: The Second Eigenvalue of the Google Matrix// Stanford University Technical Report. – 2008. – P. 20–34.
12. Google Removes Pagerank and Launches Rainbrain Technology. [Електронний ресурс]. — Режим доступу: https://web.archive.org/web/20161223012407/http://seometrics.in/google-removes-pagerank-launches-rainbrain-technology/. — Дата доступу: Травень 2019. — Назва з екрана.
13. The Periodic Table. [Електронний ресурс]. — Режим доступу: https://searchengineland.com/seotable— Дата доступу : Травень 2019. — Назва з екрана.
14. Matthew Richardson, Pedro Domingos, A. Chingos: The Intelligent Surfer:Probabilistic Combination of Link and Content Information// Cambridge, UK: Cambridge University Press. – 2001. – P. 364–374.
15. The real impact of Google’s RankBrain on search traffic. [Електронний ресурс]. — Режим доступу: https://thenextweb.com/contributors/2017/05/22/real-impact-googles-rankbrain-search-traffic/#.tnw_nYDXqPTo. — Дата доступу: Травень 2019. — Назва з екрана.
16. The Anatomy of a Large-Scale Hypertextual Web Search Engine. [Електронний ресурс]. — Режим доступу: http://infolab.stanford.edu/~backrub/google.html. — Дата доступу: Травень 2019. — Назва з екрана.
17. Google's Tensor Processing Unit could advance Moore's Law. [Електронний ресурс]. — Режим доступу: https://www.pcworld.com/article/3072256/googles-tensor-processing-unit-said-to-advance-moores-law-seven-years-into-the-future.html. — Дата доступу : Травень 2019. — Назва з екрана.

18. Google: RankBrain. [Електронний ресурс]. — Режим доступу: https://searchengineland.com/library/google/google-rankbrain. — Дата доступу : Травень 2019. — Назва з екрана.
19. What are malicious websites?. [Електронний ресурс]. — Режим доступу: https://us.norton.com/internetsecurity-malware-what-are-malicious-websites.html. — Дата доступу : Травень 2019. — Назва з екрана.
20. Best Search Engines in The World. [Електронний ресурс]. — Режим доступу: https://www.inspire.scot/blog/2016/11/11/top-12-best-search-engines-in-the-world238. — Дата доступу : Травень 2019. — Назва з екрана.
21. What is Google Team Drive. [Електронний ресурс]. — Режим доступу: https://www.systoolsgroup.com/google-drive/team-drive.html. — Дата доступу : Травень 2019. — Назва з екрана.
22. 8 major Google algorithm updates, explained. [Електронний ресурс]. — Режим доступу: https://searchengineland.com/8-major-google-algorithm-updates-explained-282627. — Дата доступу : Травень 2019. — Назва з екрана.
23. What is Yandex? A Detailed Guide to the Russian Search Engine. [Електронний ресурс]. — Режим доступу: https://detailed.com/what-is-yandex/. — Дата доступу : Травень 2019. — Назва з екрана.
24. How does a search engine work?. [Електронний ресурс]. — Режим доступу: https://www.bigcommerce.com/ecommerce-answers/how-does-search-engine-work/. — Дата доступу : Травень 2019. — Назва з екрана.
25. Sorting, searching and algorithm analysis. [Електронний ресурс]. — Режим доступу: https://python-textbok.readthedocs.io/en/1.0/Sorting_and_Searching_Algorithms.html. — Дата доступу : Травень 2019. — Назва з екрана.

26. Searching Algorithms. [Електронний ресурс]. — Режим доступу: https://www.geeksforgeeks.org/searching-algorithms/. — Дата доступу : Травень 2019. — Назва з екрана.
27. Shervin Daneshpajouh, Mojtaba Mohammadi Nasiri, Mohammad Ghodsi,: A Fast Community Based Algorithm for Generating Crawler Seeds Set// Computer Science Department, Stanford University, Stanfor. – 2008. – P. 153–178.
28. A comprehensive and scalable database search system for metaproteomics. [Електронний ресурс]. — Режим доступу: https://www.ncbi.nlm.nih.gov/pubmed/27528457. — Дата доступу : Травень 2019. — Назва з екрана.
29. Search trademark database. [Електронний ресурс]. — Режим доступу: https://www.uspto.gov/trademarks-application-process/search-trademark-database. — Дата доступу : Травень 2019. — Назва з екрана.
30. Scan Algorithms. [Електронний ресурс]. — Режим доступу: http://www.cs.ecu.edu/karl/3300/spr14/Notes/Algorithm/scan.html. — Дата доступу : Травень 2019. — Назва з екрана.
31. How Search Engines Work Using a 2 Step Process. [Електронний ресурс]. — Режим доступу: https://www.business2community.com/seo/how-search-engines-work-using-a-3-step-process-0322639. — Дата доступу : Травень 2019. — Назва з екрана.
32. Web Crawlers and User-Agents - Top 10 Most Popular. [Електронний ресурс]. — Режим доступу: https://www.keycdn.com/blog/web-crawlers. — Дата доступу : Травень 2019. — Назва з екрана.
33. Baidu search algorithms work. [Електронний ресурс]. — Режим доступу: http://research.baidu.com/Career — Дата доступу : Травень 2019. — Назва з екрана.

34. Baidu Maps Step by Step. [Електронний ресурс]. — Режим доступу: https://internchina.com/baidu-maps-how-to-use-it/ — Дата доступу : Травень 2019. — Назва з екрана.
35. Detecting Search Engines Bot & Web Spiders. [Електронний ресурс]. — Режим доступу: http://www.phacks.net/detecting-search-engine-bot-and-web-spiders/— Дата доступу : Травень 2019. — Назва з екрана.
36. Sergey Brin, Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine// Computer Science Department, Stanford University, Stanfor. – 1998. – P. 45– 48.
37. Understanding the Indexed Page on a Website. [Електронний ресурс]. — Режим доступу: https://www.lifewire.com/index-html-page-3466505— Дата доступу : Травень 2019. — Назва з екрана.
38. Search Engine Indexing. [Електронний ресурс]. — Режим доступу: https://www.deepcrawl.com/knowledge/technical-seo-library/search-engine-indexing/— Дата доступу : Травень 2019. — Назва з екрана.
39. Algolia Instant Search. [Електронний ресурс]. — Режим доступу: https://marketplace.magento.com/algolia-algoliasearch.html— Дата доступу : Травень 2019. — Назва з екрана.
40. Matrix Specification. [Електронний ресурс]. — Режим доступу: https://matrix.org/docs/spec/. — Дата доступу: Травень 2019. — Назва з екрана.
41. Kerckhoffs' principle. [Електронний ресурс]. Режим доступу: https://artofproblemsolving.com/community/c1671h1005760_kerckhoffs_principle. — Дата доступу: Травень 2019. — Назва з екрана.
42. Kelly, Diane, Jaime Teevan. Implicit feedback for inferring user preference: a bibliography// Computer Science Department, Stanford University, Stanfor. – 2003. – P. 245– 246.
43. Kolb, Whishaw. Fundamentals of Human Neuropsychology// Fundamentals of Human Neuropsychology. – 2018. – P. 453—454.

44. Mendez, Diego; Baudry, Benoit; Monperrus, Martin. IEEE 13th International Working Conference on Source Code Analysis and Manipulation// Fundamentals of Human Neuropsychology. – 2013. – P. 43—54.
45. Kerckhoffs' principle. [Електронний ресурс]. Режим доступу: https://artofproblemsolving.com/community/c1671h1005760_kerckhoffs_principle. — Дата доступу: Травень 2019. — Назва з екрана.
46. Hopmann, Alex. Story of XMLHTTP// Alex Hopmann's Blog. – 2010. – P. 233—244.
47. Operating System | Thread. [Електронний ресурс]. — Режим доступу: https://www.geeksforgeeks.org/operarting-system-thread/ — Дата доступу: Травень 2019. — Назва з екрана.
48. The MongoDB. [Електронний ресурс]. — Режим доступу: https://docs.mongodb.com/manual/ — Дата доступу: Травень 2019. — Назва з екрана.
49. The MongoDB in deep learn. [Електронний ресурс]. — Режим доступу: https://docs.mongodb.com/manual/tutorial/ — Дата доступу: Травень 2019. — Назва з екрана.
50. Universally Unique Identifiers. [Електронний ресурс]. — Режим доступу: https://www.itu.int/en/ITU-T/asn1/Pages/UUID/uuids.aspx — Дата доступу: Травень 2019. — Назва з екрана.
51. TECH MEDIA COMPANY IN THE WORLD. [Електронний ресурс]. — Режим доступу: https://www.idg.com/advertise-with-us/media-brands/media-region/regions/ — Дата доступу: Травень 2019. — Назва з екрана.
52. DATAGROUP COMPANY HAS BECOME AN INTERNET. [Електронний ресурс]. — Режим доступу: https://www.datagroup.ua/en/novyny/kompaniya-datagrup-stala-internet-partnerom-mizhnarodnogo-it-130 — Дата доступу: Травень 2019. — Назва з екрана.

ДОДАТКИ

ДОДАТОК 1

Текст програми

Лістинг 1. Фрагмент тексту detectedQuery для аналізу запиту

```
import startsWith from "lodash/startsWith";
import endsWith from "lodash/endsWith";
import isEmpty from "lodash/isEmpty";

const operatorVS = (query, elements) => {
  const names = elements.map((elem) => elem.name);
  const joinNames = names.join("|");

  let SCheck = { id: "", state: false };
  let ECheck = { id: "", state: false };

  elements.forEach(({ name, id }) => {
    if (startsWith(query, name)) {
      SCheck.id = id;
      SCheck.state = true;
    }
    if (endsWith(query, name)) {
      ECheck.id = id;
      ECheck.state = true;
    }
  });

  if (
    query.search(new RegExp(`(${joinNames}) vs (${joinNames})`)) == 0 &&
    ECheck.state &&
    SCheck.state
  ) {
    return { S: SCheck.id, E: ECheck.id };
  } else {
    return false;
  }
};

const findName = (query, elements) => {
  let find = false;

  elements.forEach(({ name, id }) => {
    if (startsWith(query, name) && endsWith(query, name)) {
      find = id;
    }
  });
```

```
  return find;
};

const findKey = (query, elements) => {
  const values = elements[0].values.map(({ name, id }) => ({ name, id }));
  let find = false;

  values.forEach(({ name, id }) => {
    if (startsWith(query, name) && endsWith(query, name)) {
      find = id;
    }
  });

  return find;
};

export const detectedQuery = (query, elements) => {
  const VS = operatorVS(query, elements);
  const fName = findName(query, elements);
  const fKey = findKey(query, elements);

  console.log("elements: ", elements);

  if (!isEmpty(VS)) {
    const findElements = [];
    findElements.push(elements.find(({ id }) => id == VS.S));
    findElements.push(elements.find(({ id }) => id == VS.E));

    return {
      type: "vs",
      findElements,
      render: true,
    };
  }

  if (!isEmpty(fName)) {
    const findElements = [];
    findElements.push(elements.find(({ id }) => id == fName));

    return {
      type: "fineName",
      findElements,
      render: true,
```

```
    };
  }

  if (!isEmpty(fKey)) {
    return {
      type: "findKey",
      findElements: fKey,
      render: true,
    };
  }

  return { type: "", render: false };
};
```

Лістинг 2. Фрагмент тексту ListSearchServices для ранжування сервісів

```
import React from "react";
import { observer } from "mobx-react-lite";
import { useStore } from "../store";
import styled from "styled-components/macro";
import { useTransition, animated } from "react-spring";
import { FuseSettingsServices } from "../utils";
import Fuse from "fuse.js";
import isEmpty from "lodash/isEmpty";
import { t, date } from "@lingui/macro";
import { i18n } from "../utils";
import { withRouter } from "react-router";

const Container = styled.div`
  display: flex;
  flex-direction: column;
  align-items: center;

  margin: 32px;
`;

const Card = styled(animated.div)`
  display: flex;
  flex-direction: column;
```

```
  padding: 16px;
  margin: 8px;
  box-shadow: 0 3px 6px rgba(0, 0, 0, 0.16), 0 3px 6px rgba(0, 0, 0,
0.23);

  width: 60%;
  height: 120px;

  cursor: pointer;
`;

const CardTitle = styled.div`
  ${({ theme }) => theme.typography({ variant: "miniTitle" })}
`;

const CardDescription = styled.div`
  ${({ theme }) => theme.typography({ variant: "body" })}
  font-weight: 600;
`;

const CardLeftDescription = styled.div`
  ${({ theme }) => theme.typography({ variant: "body" })}
  font-weight: 600;
  margin-left: auto;
`;

const CardLiner = styled.div`
  display: flex;

  margin-top: auto;
`;

const ListSearchServices = observer(
  withRouter(({ userQuery, history }) => {
    const { Services } = useStore();
    const SearchServicesView = [];

    if (userQuery !== "") {
      Services.entries.forEach((service) => {
```

```
    SearchServicesView.push(service);
  });
}

const fuse = new Fuse(SearchServicesView, FuseSettingsServices);
const SearchServicesFilteredView = fuse.search(userQuery);

const transitions = useTransition(
  SearchServicesFilteredView,
  (item) => item.id,
  {
    from: { opacity: 0.1 },
    enter: { opacity: 1 },
    leave: { opacity: 0.1 },
    trail: 50,
  },
);

return (
  <Container>
    {userQuery !== "" ? (
      isEmpty(transitions) ? (
        <CardTitle>LOL KEK</CardTitle>
      ) : (
        transitions.map(
          ({
            item: { createdAt, id, description, name, author },
            props,
            key,
          }) => {
            return (
              <Card
                key={key}
                style={props}
                onClick={() => {
                  history.push(`/service/${id}`);
                }}
              >
                <CardTitle>{name}</CardTitle>
```

```
                    <CardDescription>{description}</CardDescription>
                    <CardLiner>
                      <CardDescription>Author:
{author}</CardDescription>
                      <CardLeftDescription>
                        {i18n._(t`${date(createdAt)}`)}
                      </CardLeftDescription>
                    </CardLiner>
                  </Card>
                );
              },
            )
          )
        ) : (
          <CardTitle>This list is Empty please start
search</CardTitle>
        )}
      </Container>
    );
  }),
);

export default ListSearchServices;
```

Лістинг 3. Фрагмент тексту selectService для створення бази даних

```
import { flow, getRoot, types, applySnapshot } from "mobx-state-
tree";
import { req } from "../utils";

const Keys = types.model("keys", {
  id: types.maybeNull(types.string),
  name: types.maybeNull(types.string),
  type: types.maybeNull(types.string),
  unit: types.maybeNull(types.string),
});

const Value = types.model("value", {
```

```
  id: types.maybeNull(types.string),
  value: types.maybeNull(types.string),
});

const Elements = types.model("elements", {
  id: types.maybeNull(types.string),
  name: types.maybeNull(types.string),
  img: types.maybeNull(types.string),
  description: types.maybeNull(types.string),
  values: types.optional(types.array(Value), []),
});

const FullDescription = types.model("fullDescription", {
  title: types.maybeNull(types.string),
  body: types.maybeNull(types.string),
  img: types.maybeNull(types.string),
});

const selectService = types
  .model("selectService", {
    id: types.maybeNull(types.string),
    name: types.maybeNull(types.string),
    description: types.maybeNull(types.string),
    createdAt: types.maybeNull(types.string),
    author: types.maybeNull(types.string),
    keys: types.optional(types.array(Keys), []),
    elements: types.optional(types.array(Elements), []),
    fullDescription: types.optional(FullDescription, {}),
  })
  .actions((self) => ({
    fetchService: flow(function*(ServiceId) {
      const token = getRoot(self).User.guestToken;

      const res = yield req("services_data", {
```

```
        method: "GET",
        headers: {
          "content-type": "application/json",
          Authorization: token,
        },
        searchParams: {
          id: ServiceId,
        },
      }).json();

      applySnapshot(self, res);
    }),

    getFullDescription() {
      return {
        title: self.fullDescription.title,
        body: self.fullDescription.body,
        img: self.fullDescription.img,
      };
    },

    getElements() {
      return self.elements.map(({ id, values, name, img, description
}) => ({
        values: values.map(({ id, value }) => {
          const findKey = self.keys.find((key) => key.id === id);

          return { id, value, ...findKey };
        }),
        name,
        img,
        description,
        id,
      }));
```

```
    },
  }));

export default selectService;
 import store from "./store";
import pick from "lodash/pick";
import isEmpty from "lodash/isEmpty";
import find from "lodash/find";
import uuid from "uuid/v4";
import { DateTime } from "luxon";

export const getUsers = (filter = []) => {
  const Users = store.get("Users").value();

  if (isEmpty(filter)) return SearchServices;

  return Users.map(User => pick(User, filter));
};

export const addUser = (name, description, login, password) => {
  if (
    isEmpty(name) ||
    isEmpty(description) ||
    isEmpty(login) ||
    isEmpty(password)
  ) {
    console.error("Error addUser fields is empty");
    return false;
  }

  store
    .get("Users")
    .push({
      id: uuid(),
```

```
      token: uuid(),
      createdAt: DateTime.local().toISO(),
      name,
      description,
      login,
      password
    })
    .write();

  return true;
};

export const getUserByToken = (token, filter = []) => {
  if (isEmpty(token)) {
    console.error("Error getUserByToken token is empty");
    return false;
  }

  const Users = store.get("Users").value();
  const foundUser = find(Users, User => {
    return User.token == token;
  });

  if (isEmpty(foundUser)) {
    console.error("Error getUserByToken foundUser is empty");
    return false;
  }

  if (isEmpty(filter)) return foundUser;

  return pick(foundUser, filter);
};
```

ДОДАТОК 2

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Метод динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувачів

Науковий керівник: к.т.н., Олещенко Любов Михайлівна

Студент: Коломієць Іван Валерійович

Київ – 2019

АКТУАЛЬНІСТЬ

НАУКОВЕ ЗАВДАННЯ

Удосконалити програмні методи пошуку інформації в мережі Інтернет

МЕТА ДОСЛІДЖЕННЯ

Збільшити швидкість виконання ранжування пошукової видачі та зменшити витрати дискового простору пошукової системи за рахунок створення удосконаленого методу динамічного пошуку з урахуванням індивідуальних особливостей користувачів

ОБ'ЄКТ ДОСЛІДЖЕННЯ

- процес програмної організації пошуку інформації в мережі Інтернет

ПРЕДМЕТ ДОСЛІДЖЕННЯ

- методи інформаційного пошуку на основі адаптації процесу пошуку до індивідуальних особливостей користувачів інформаційно-пошукових систем

ПРОБЛЕМАТИКА

Розроблені на даний момент методи інформаційного пошуку недостатньою мірою враховують специфіку пошукових запитів користувачів мережі Інтернет.

Такому пошуку притаманна низька вартість зв’язку, неоднорідність і різноманітність інформаційних ресурсів.

ЗАДАЧІ

1. Аналіз існуючих методів пошуку інформації сучасних інформаційно-пошукових систем (ІПС)

2. Формулювання критеріїв для створення удосконаленого методу динамічного пошуку з урахуванням індивідуальних особливостей користувачів ІПС

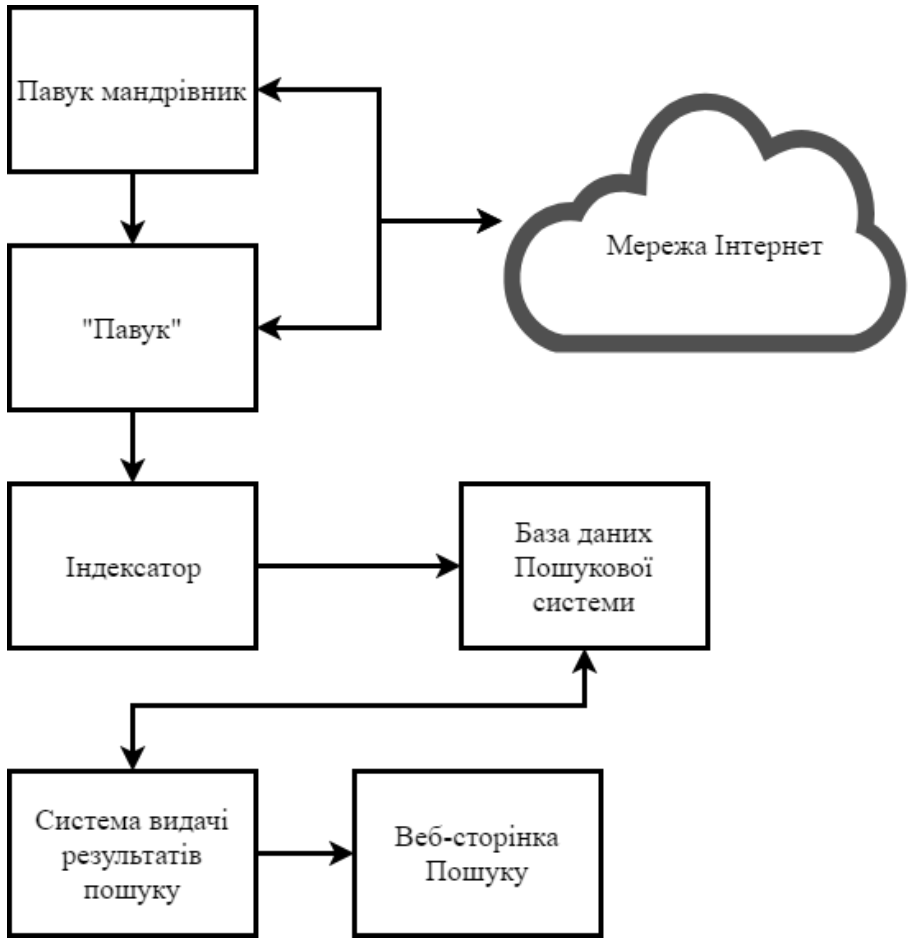
3. Вибір стеку технологій для проектування ІПС

4. Оцінка ефективності запропонованого методу

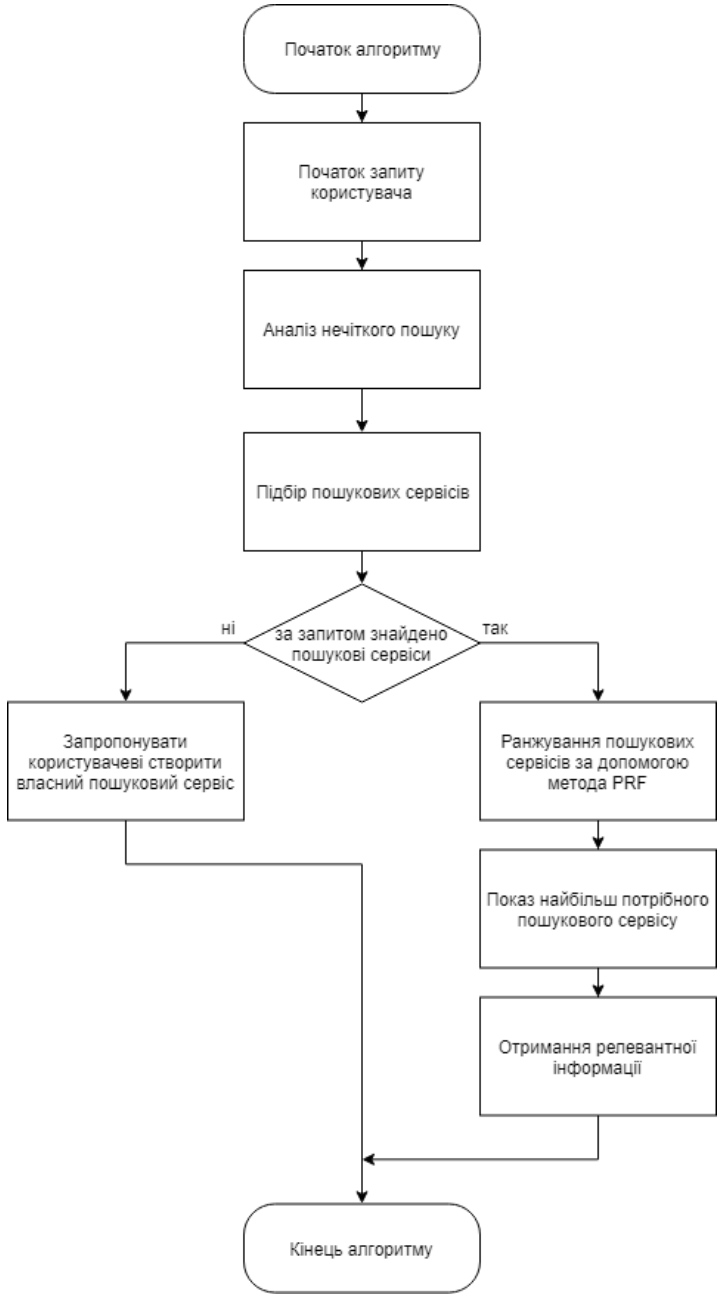
Проблеми сучасних пошукових сервісів

- Алгоритм «Індексатор»
- Алгоритм «Павук»
- Метадані веб-сторінок

Проблеми сучасних пошукових систем



Запропонований метод індивідуального пошуку



Показник релевантності

$$Score = \frac{W_{single} + W_{pair} + \frac{1}{k} W_{AllWords} + kW_{Phrase} + W_{PRF}}{k}$$

- $\boldsymbol{W_{single}}$ – збіг слова із запиту в пошуковій системі
- $\boldsymbol{W_{pair}}$ – збіг пар слів із запиту
- $\boldsymbol{W_{AllWords}}$ – складова, що дає пріоритет за наявністю всіх слів запиту в пошуковій системі
- $\boldsymbol{W_{Phrase}}$ – збіг фраз запиту
- $\boldsymbol{W_{PRF}}$ – Score попередніх ітерацій

Складові показника релевантності

$$W_{pair} = \sum_{j=0}^{countQueryWords} \sum_{i=0}^{countWords} SelectWord_j * SingleWord_i \ ,$$

$$W_{prime} = [PrimeWord_0, PrimeWord_{1,\dots,} PrimeWord_{countWords}],$$

$$W_{single} = \sum_{i=0}^{countWords} ScoreWord_i * PrimeWord_i,$$

$$W_{allWorlds} = \sum_{i=0}^{countWords} ScoreWord_i * SelectWord_i,$$

$$W_{phrase} = \sum_{i=0}^{countWords} W_{allWorlds} * QueryWorlds$$

Складові показника релевантності

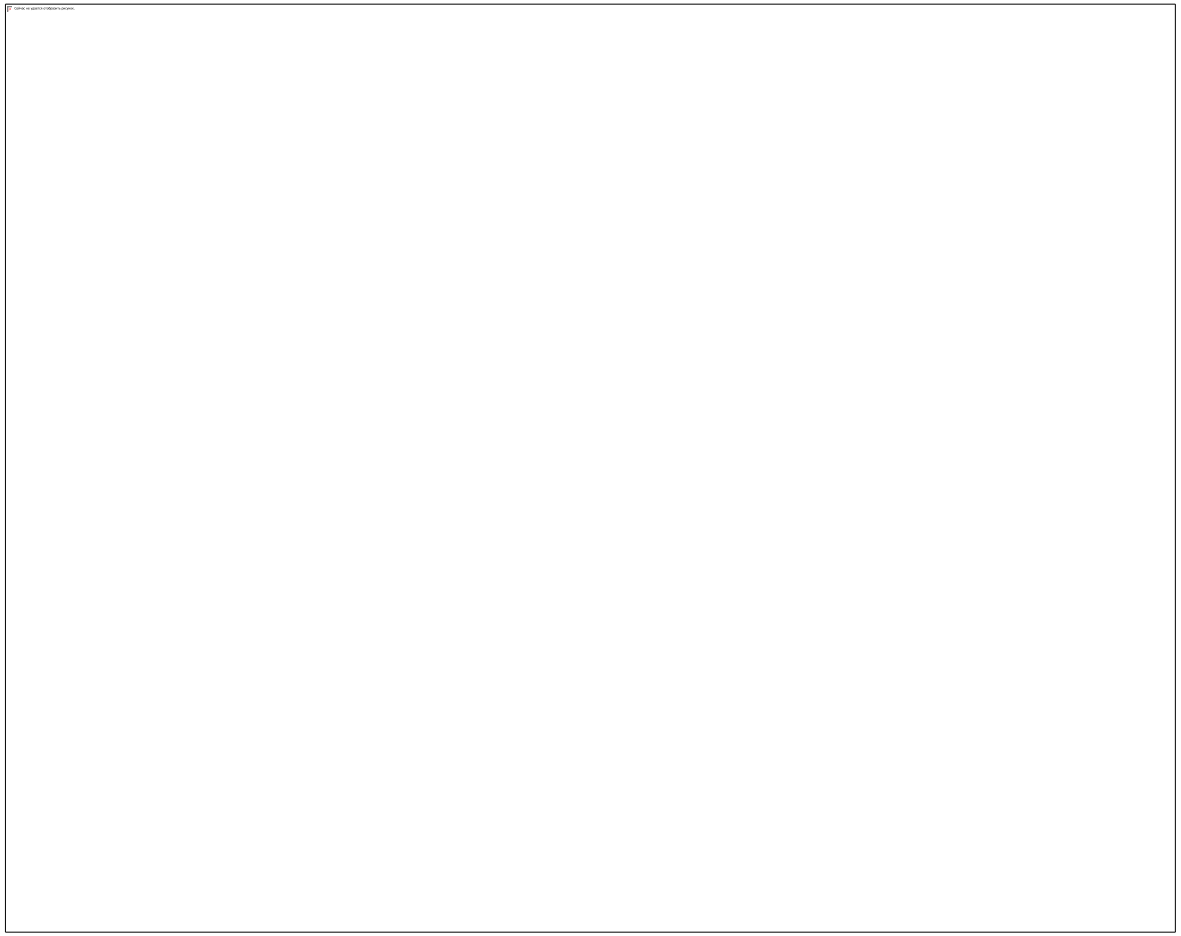
$$RatingVocabulary = [ScoreWord_0, \dots, ScoreWord_{countWords}],$$

$$Average_{RV} = \frac{1}{countWords} \sum_{i=0}^{countWords} ScoreWord_i,$$

$$Score_{wordQ} = [ScoreQ_0, ScoreQ_{1,\dots,} ScoreQ_{countQWords}],$$

$$ScoreQ = \left[\frac{ScoreQ_{lastmonth}}{Range_{last}}\right] Range_{last} + \frac{1}{2^4} Range_{quarter} + \frac{1}{2^7} Range_{year},$$

Особливості архітектури програмного комплексу



Використанні технології

ЗАГАЛЬНІ

- Система контролю версій — GitLab

СЕРВЕР ТА API

- Мова програмування — NodeJS
- Спілкування з API — протоколи HTTPS, OAuth 2.0, та технологія WebHooks

КЛІЄНТСЬКА ЧАСТИНА

- Мова програмування — JavaScript
- UI — ReactJS
- Пакувальник — Parcel
- Транспілятор — Babel
- Управління станом додатку — MST
- Навигація по додатку — React Router

БАЗА ДАНИХ

- Інтерфейс бази — Mongodb
- Формат даних — JSON

Користувачі ІПС

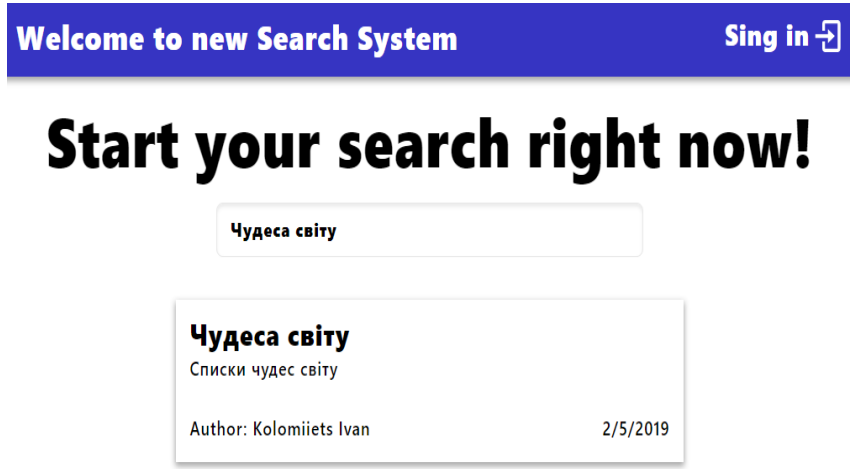
Розробник пошукових сервісів

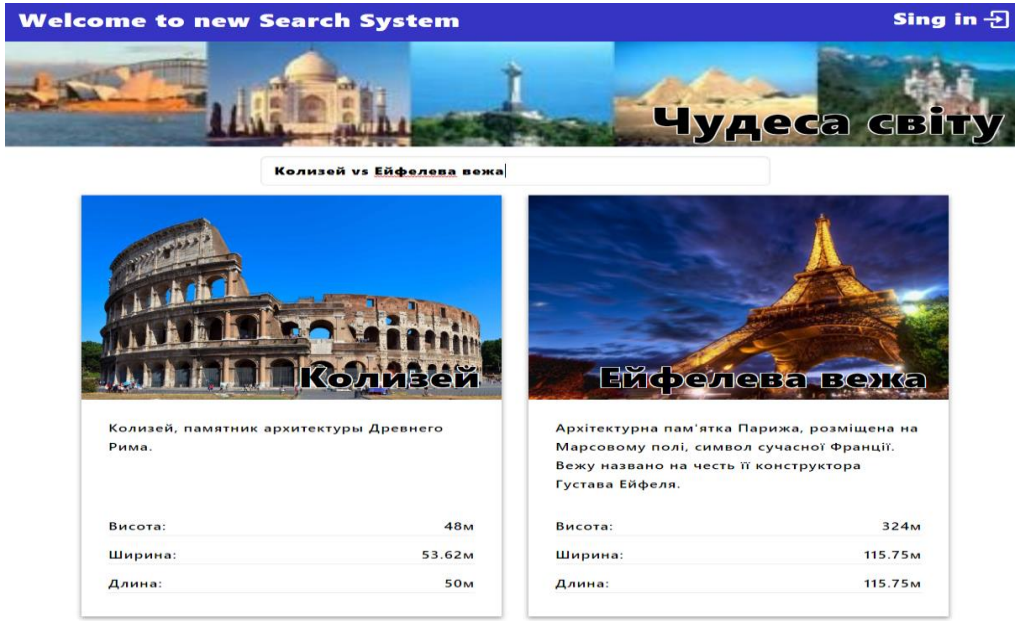
- Створює власний пошуковий сервіс
- Заповнює пошуковий сервіс інформацією
- Робить оновлення інформації

Кінцевий користувач

- Здійснює вибір потрібного пошукового сервісу
- Використовує пошуковий сервіс для роботи з інформацією

Адаптація процесу пошуку до індивідуальних особливостей користувачів





$$Score = \frac{W_{single} + W_{pair} + \frac{1}{k} W_{AllWords} + kW_{Phrase} + W_{PRF}}{k}$$

НАУКОВА НОВИЗНА

Запропоновано метод, який, на відміну існуючих методів пошуку інформації в мережі Інтернет, дозволяє збільшити швидкість виконання ранжування пошукової видачі на 10.2%.

Запропоновано новий підхід для пошуку інформації, який дозволяє зменшити витрати дискового простору пошукової системи на 17.61% за рахунок відмови від індексування веб-сторінок, яке використовується в сучасних пошукових системах.

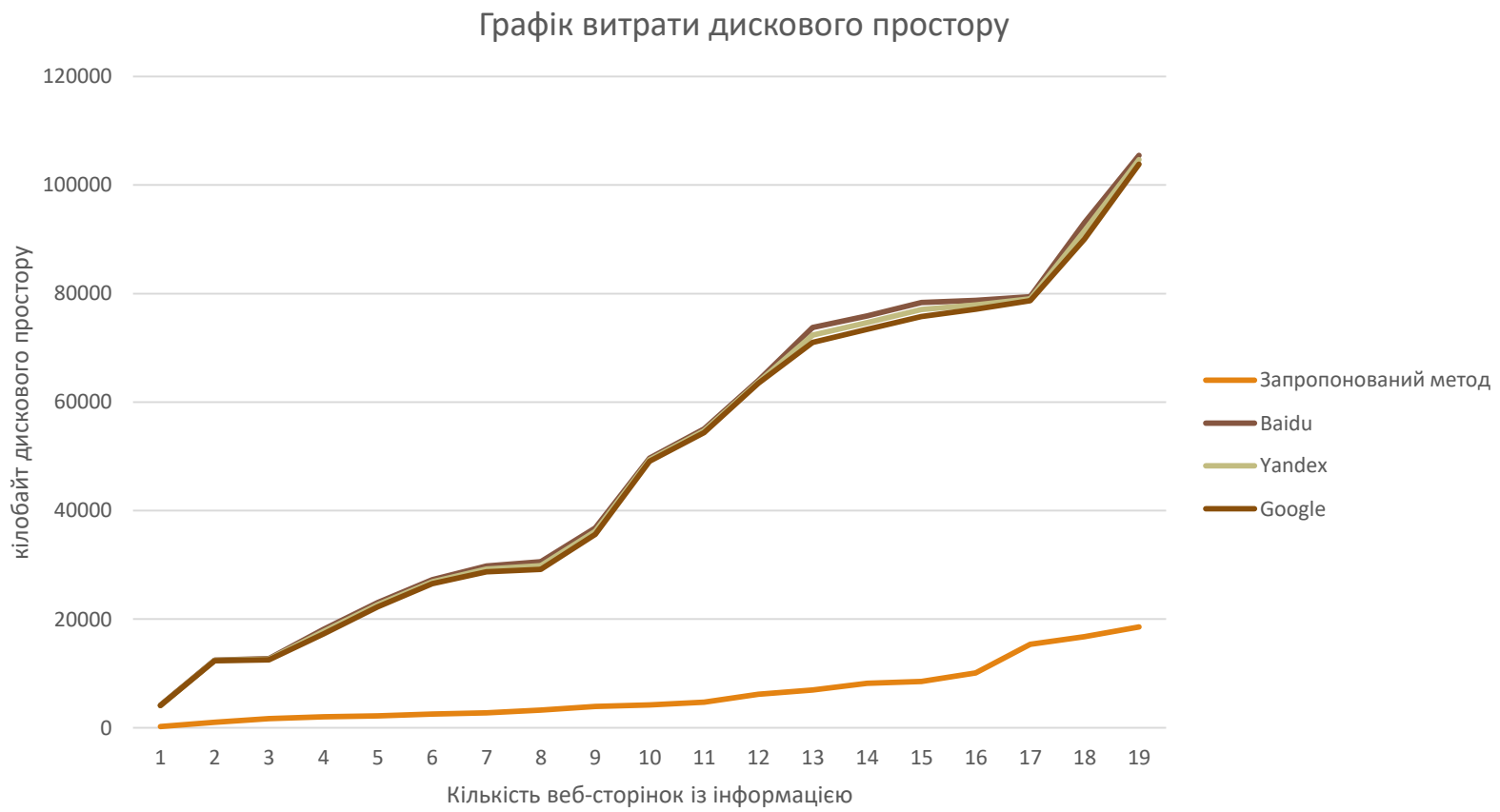
Порівняння аспектів індексування пошукових систем

Назва пошукової системи	Швидкість індексації в годинах	Оновлення індексації в годинах
Запропонований метод	Від 0,4	Від 0,1
Google	Від 18	Від 0.3 до 48
Yandex	Від 22	Від 0.1 до 24
Baidu	Від 4	Від 1 до 24

Результати тестування навантаження системи

Запропонований метод	Надання інформації в мс	Швидкість відповіді із серверу в мс	Швидкість ранжування в мс	Загалом в мс
Запропонований метод	448	710	2683	3 841
Google	463	200	3372	4 035
Yandex	378	250	3183	3 811
Baidu	446	350	3781	4 577

Графік витрат дискового простору



ВИСНОВКИ

1. Здійснено аналіз існуючих методів пошуку інформації сучасних інформаційно-пошукових систем та виділено їх основні недоліки.

2. На основі виділених недоліків методів пошуку сучасних ІПС сформульовано критерії для створення удосконаленого методу динамічного пошуку з урахуванням індивідуальних особливостей користувачів ІПС.

3. Запропоновано програмну платформу для створення власних пошукових сервісів. Використано ранжування з використанням PRF метод.

4. Розроблений метод динамічного проектування пошукового сервісу з урахуванням індивідуальних особливостей користувачів дозволяє збільшити швидкість виконання ранжування пошукової видачі на 10.2% та зменшити витрати дискового простору пошукової системи на 17.61%.

5. Подальшим розвитком даного методу є….

Апробація

1. Kolomiets I.V., Oleshchenko L.M. The Method of Dynamic Design of a Search Engine Based on Automated Analysis of User Requests // ISSN 1990-5548 (фахове видання) Electronics and Control Systems 2018. N4(58): 77-82.

2. Олещенко Л.М., Коломієць І.В. Метод динамічного проектування пошукового сервісу на основі адаптації процесу пошуку до індивідуальних особливостей користувача // ПМК-2018 (2). – С. 217-221.

Дякую за увагу!